Two-Dimensional RC/SW Constrained Codes: Bounded Weight and Almost Balanced Weight

Tuan Thanh Nguyen, Kui Cai, *Senior Member, IEEE*, Han Mao Kiah, *Senior Member, IEEE*, Kees A. Schouhamer Immink, *Life Fellow, IEEE*, and Yeow Meng Chee, *Senior Member, IEEE*

***Abstract*— In this work, we study two types of constraints on two-dimensional binary arrays. Given $p \in [0,1], \epsilon \in [0,1/2]$, we study 1) the p-bounded constraint: a binary vector of size n is said to be p-bounded if its weight is at most pn, and 2) the ϵ-balanced constraint: a binary vector of size n is said to be ϵ-balanced if its weight is within $\big[(1/2-\epsilon)n,(1/2+\epsilon)n\big]$. Such constraints are crucial in several data storage systems, those regard the information data as two-dimensional (2D) instead of one-dimensional (1D), such as the crossbar resistive memory arrays and the holographic data storage. In this work, efficient encoding/decoding algorithms are presented for binary arrays so that the weight constraint (either p-bounded constraint or ϵ-balanced constraint) is enforced over every row and every column, regarded as 2D row-column (RC) constrained codes; or over every window (where each window refers to as a sub-array consisting of consecutive rows and consecutive columns), regarded as 2D sliding-window (SW) constrained codes. While low-complexity designs have been proposed in the literature, mostly focusing on 2D RC constrained codes where $p = 1/2$ and $\epsilon = 0$, this work provides efficient coding methods that work for both 2D RC constrained codes and 2D SW constrained codes, and more importantly, the methods are applicable for arbitrary values of p and ϵ. Furthermore, for certain values of p and ϵ, we show that, for sufficiently large array size, there exists linear-time encoding/decoding algorithm that incurs at most one redundant bit.**

***Index Terms*— Two-dimensional (2D) constrained codes, sliding-window (SW) constrained codes, balanced codes, almost-balanced codes, bounded weight codes, encoding, decoding.**

Manuscript received 18 August 2022; revised 22 December 2022; accepted 5 February 2023. Date of publication 14 February 2023; date of current version 14 July 2023. The work of Tuan Thanh Nguyen and Kui Cai was supported by the Singapore Ministry of Education Academic Research Funds Tier 2 under Grant MOE2019-T2-2-123 and Grant T2EP50221-0036. The work of Han Mao Kiah was supported by the Ministry of Education, Singapore, under its MOE AcRF Tier 2 Award under Grant MOE-T2EP20121-0007. An earlier version of this paper was presented in part at the 2021 IEEE International Symposium on Information Theory [DOI: 10.1109/ISIT45174.2021.9517970] and in part at the 2022 IEEE International Symposium on Information Theory [DOI: 10.1109/ISIT50566.2022.9834724]. *(Corresponding author: Tuan Thanh Nguyen.)*

Tuan Thanh Nguyen and Kui Cai are with the Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, Singapore 487372 (e-mail: tuanthanh_nguyen@sutd.edu.sg; cai_kui@sutd.edu.sg).

Han Mao Kiah is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371 (e-mail: hmkiah@ntu.edu.sg).

Kees A. Schouhamer Immink is with Turing Machines Inc., 3016 DK Rotterdam, The Netherlands (e-mail: immink@turing-machines.com).

Yeow Meng Chee is with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 119077 (e-mail: ymchee@nus.edu.sg).

Communicated by G. Ge, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2023.3244979.

Digital Object Identifier 10.1109/TIT.2023.3244979

I. Introduction

TWO-DIMENSIONAL (2D) weight-constrained codes have attracted recent attention due to various applications in modern storage devices that are attempting to increase the storage density by regarding the information data as two-dimensional binary arrays. In this work, we are motivated by the application of 2D weight-constrained codes in the *holographic recording systems* and the *resistive memory* based on crossbar arrays. In particular, in optical recording, the holographic memory capitalizes on the fact that the recording device is a surface, and therefore the recording data should be regarded as 2D, as opposed to the track-oriented one-dimensional (1D) recording paradigm [3], [4], [5], [6]. On the other hand, the key in resistive memory technologies is that the memory cell is a passive two-terminal device that can be both read and written over a simple crossbar structure [7], [8], [9], [10]. Both models offer a huge density advantage, however, face new reliability issues and introduce new types of constraints, which are now 2D constraints, rather than 1D constraints. We next briefly describe the motivation of our study on the two types of 2D weight constraints: the *bounded weight constraint* and the *almost-balanced weight constraint*.

The 2D bounded weight constraint is used to limit the number of 1's in an array. It has been suggested as an effective method to reduce the *sneak path* effect, a fundamental and challenging problem, in the crossbar memory arrays. The sneak path problem was addressed by numerous works with different approaches and at various system layers [11], [12], [13], [14], [15], [16]. In particular, when a cell in a crossbar array is read, a voltage is applied upon it, and current measurement determines whether it is in a low-resistance state (LRS, corresponding to a '1') or a high-resistance state (HRS, corresponding to a '0'). The sneak path occurs when a resistor in the HRS is being read, current also passes through a series of resistors in the LRS that exists in parallel to it, thereby causing it to be erroneously read as low resistance. Therefore, by enforcing fewer memory cells with the LRSs, we can reduce the sneak path effect. This can be achieved by applying constrained coding techniques to convert the user data into a 2D-constrained array that limits the number of 1's. Motivated by the application, we extend the study of 2D weight-constrained codes in the literature works in two variations:

- Design of 2D p-bounded row-column (RC) constrained codes over $n \times n$ arrays, that limit the number of 1's in every row and every column to be at most pn. For a specific value of p, Ordentlich and Roth [11] required the weight in every row and every column to be at most half, i.e. $p = 1/2$, and presented efficient encoders with redundancy at most $2n$ for $n \times n$ arrays. In [17], the authors studied the bounds of codes that require the weight in every row and every column to be precisely pn and provided a coding scheme based on the *enumeration coding technique*. In this work, we extend the study of 2D p-bounded RC constrained codes and present efficient encoding/decoding algorithms for arbitrary $p \in [0, 1]$.
- Design of 2D p-bounded sliding-window (SW) constrained codes over $n \times n$ arrays, that limit the number of 1's in every window of size $m \times m$ for some $m \leqslant n$. It was proved in [13] that imposing constraints to completely eliminate sneak paths in the array implies storage capacity that vanishes with the array dimensions. Several attempts have been made to avoid sneak paths without ending up with zero capacity (for example, see [13], [14], [16]). A common idea is to divide the array rows into *disjoint subsets*, where each is a subarray consisting of consecutive rows and columns, and then study the coding method required to prevent sneak paths over these subarrays. Particularly, in [16], the authors constructed arrays of size $n\times n$ by concatenating subarrays of size $m\times m$ for $m \leqslant 4$. In this work, we enforce the p-bounded weight constraint over every window (where each window refers to as a subarray consisting of consecutive rows and consecutive columns), regarded as 2D sliding-window (SW) constrained codes. Since the windows are not necessarily disjoint, the design of constrained codes over every window is more challenging than over disjoint subarrays consisting of consecutive rows and consecutive columns.

On the other hand, the 2D almost-balanced weight constraint is used to control the imbalance between '1's and '0's in an array. This constraint is crucial in holographic recording systems. In such systems, data is stored optically in the form of 2D pages [5], [6], and each data page is a pattern of '0's and '1's, represented by *dark* and *light* spots, respectively. To improve the reliability of the holographic recording system, one suggested solution by Vardy et al. [18] is to use coding techniques that do not permit a large imbalance between '0's and '1's so that, during recording, the amount of signal light be independent of the data content. In [17], Ordentlich and Roth also emphasized the importance of balanced codes, and experiments' reports on holographic memory and other existing optical devices also suggested that '0's and '1's in the recorded data need to be balanced within certain areas or patterns. Motivated by the application, in this work, we study the 2D ϵ-*balanced* weight constraint defined over $n\times n$ arrays via two different models: the 2D ϵ-balanced RC constrained codes, that enforce every row and every column to be ϵ-balanced; and the 2D ϵ-balanced m-SW constrained codes, that enforce every window of size $m \times m$ to be ϵ-balanced for some $m \leqslant n$. For 2D ϵ-balanced RC constrained codes, when $\epsilon = 0$ and n is even, Talyansky et al. [3] enforced the weight in every row and every column of $n \times n$ array to be exactly $n/2$ and presented an efficient encoding method, that uses roughly $2n \log n + \Theta(n \log\log n)$ redundant bits. To further reduce the redundancy, instead of using one of the algorithms in [19] and [20], Talyansky et al. balanced the rows with the (more computationally complex) enumerative coding technique. Consequently, the redundancy can be reduced to $(3/2)n \log n + \Theta(n \log\log n)$ redundant bits. On the other hand, there is no known design for an arbitrary value of ϵ. The efficient design of 2D ϵ-balanced RC constrained codes and 2D ϵ-balanced SW constrained codes, given an arbitrary value of $\epsilon > 0$, is the main contribution of this work.

In summary, we present two efficient encoding methods for 2D RC constrained codes and 2D SW constrained codes for arbitrary $p \in [0, 1]$ and $\epsilon \in [0, 1/2]$. The coding methods are based on: (method A) the divide and conquer algorithm and a modification of the Knuth's balancing technique, and (method B) the sequence replacement technique. The coding rate of our proposed methods approaches the channel capacity for all p, ϵ. In addition, for certain values of p and ϵ, we show that for sufficiently large n, method B incurs at most one redundant bit.

We first go through certain notations and review prior-art coding techniques.

II. Preliminaries

A. Notations

Given two binary sequences $\boldsymbol{x} = x_1 \ldots x_{n_1}$ and $\boldsymbol{y} = y_1 \ldots y_{n_2}$, the *concatenation* of the two sequences is defined by $\boldsymbol{xy} \triangleq x_1 \ldots x_{n_1} y_1 \ldots y_{n_2}$. For a binary sequence $\boldsymbol{x}$, we use $\text{wt}(\boldsymbol{x})$ to denote the weight of $\boldsymbol{x}$, i.e the number of ones in $\boldsymbol{x}$. We use $\overline{\boldsymbol{x}}$ to denote the complement of $\boldsymbol{x}$. For example, if $\boldsymbol{x} = 00111$ then $\text{wt}(\boldsymbol{x}) = 3$ and $\overline{\boldsymbol{x}} = 11000$.

Let $\boldsymbol{A}_n$ denote the set of all $n \times n$ binary arrays. The ith row of an array $A \in \boldsymbol{A}_n$ is denoted by A_i and the jth column is denoted by A^j. For some $m \leqslant n$, a subarray W of size $m \times m$ of A is called a *window* if W consists of m consecutive rows and m consecutive columns. Note that an $n \times n$ binary array A can be viewed as a binary sequence of length n^2. We define $\Phi(A)$ as a binary sequence of length n^2 where bits of the array A are read row by row.

Example 1: Given $n = 3, m = 2$, consider the array

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}, \text{ then we have } \Phi(A) = abcdefghi.$$

In addition, we have four windows of size 2×2 of A as

$$W_1 = \begin{pmatrix} a & b \\ d & e \end{pmatrix}, W_2 = \begin{pmatrix} b & c \\ e & f \end{pmatrix},$$

$$W_3 = \begin{pmatrix} d & e \\ g & h \end{pmatrix}, W_4 = \begin{pmatrix} e & f \\ h & i \end{pmatrix}.$$

Definition 1: Given n, ϵ where $\epsilon \in [0, 1/2]$ and n is even, a binary sequence $\boldsymbol{x}$ of length n is said to be ϵ-balanced if

$\mathrm{wt}(\boldsymbol{x}) \in \big[(1/2-\epsilon)n, (1/2+\epsilon)n\big]$. When $\epsilon = 0$, we say the sequence is balanced. Similarly, a binary array of size $n_1 \times n_2$ is said to be ϵ-balanced if the number of 1's in the array is within $\big[(1/2-\epsilon)n_1n_2, (1/2+\epsilon)n_1n_2\big]$.

Definition 2: Given n, p, where $p \in [0,1]$, a binary sequence $\boldsymbol{x}$ of length n is said to be p-bounded if $\mathrm{wt}(\boldsymbol{x}) \leqslant pn$. Similarly, a binary array of size $n_1 \times n_2$ is said to be p-bounded if the number of 1's in the array is at most pn_1n_2.

We now show that it suffices to restrict the additional condition that pn and ϵn are integers.

Proposition 1: Given $n > 0, p \in [0,1]$, set $p' = \lfloor np \rfloor / n$. For arbitrary $\boldsymbol{x} \in \{0,1\}^n$, $\boldsymbol{x}$ is p-bounded if and only if $\boldsymbol{x}$ is p'-bounded. Similarly, given $n > 0$, n is even, and $\epsilon \in [0, 1/2]$, set $\epsilon' = \lfloor \epsilon n \rfloor / n$. For arbitrary $\boldsymbol{x} \in \{0,1\}^n$, we then have $\boldsymbol{x}$ is ϵ-balanced if and only if $\boldsymbol{x}$ is ϵ'-balanced.

Proof: We first show that for $\boldsymbol{x} \in \{0,1\}^n$, $\boldsymbol{x}$ is p-bounded if and only if $\boldsymbol{x}$ is p'-bounded.

- Assume that $\boldsymbol{x}$ is p-bounded. By definition, we have $\mathrm{wt}(\boldsymbol{x}) \leqslant pn$. Since $\mathrm{wt}(\boldsymbol{x})$ is an integer, we conclude that $\mathrm{wt}(\boldsymbol{x}) \leqslant \lfloor pn \rfloor = (\lfloor pn \rfloor / n) \times n = p'n$. Hence, $\boldsymbol{x}$ is p'-bounded.
- On the other hand, suppose that $\boldsymbol{x}$ is p'-bounded, i.e. we have $\mathrm{wt}(\boldsymbol{x}) \leqslant p'n$. Since $p' = \lfloor np \rfloor / n \leqslant np/n = p$, it implies $\mathrm{wt}(\boldsymbol{x}) \leqslant p'n \leqslant pn$. Hence, $\boldsymbol{x}$ is p-bounded.

Similarly, we can show that $\boldsymbol{x}$ is ϵ-balanced if and only if $\boldsymbol{x}$ is ϵ'-balanced.

- Assume that $\boldsymbol{x}$ is ϵ-balanced. By definition, we have $\mathrm{wt}(\boldsymbol{x}) \in \big[(1/2-\epsilon)n, (1/2+\epsilon)n\big]$. Since $\mathrm{wt}(\boldsymbol{x})$ is an integer and n is even, we conclude that
$$\begin{aligned} \lceil (1/2-\epsilon)n \rceil &\leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant \lfloor (1/2+\epsilon)n \rfloor, \text{ or} \\ n/2 - \lfloor \epsilon n \rfloor &\leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n/2 + \lfloor \epsilon n \rfloor, \text{ or} \\ n/2 - (\lfloor \epsilon n \rfloor / n) \times n &\leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n/2 + (\lfloor \epsilon n \rfloor / n) \times n, \text{ or} \\ (1/2-\epsilon')n &\leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant (1/2+\epsilon')n. \end{aligned}$$
In other words, it implies that $\boldsymbol{x}$ is also ϵ'-balanced.
- On the other hand, suppose that $\boldsymbol{x}$ is ϵ'-balanced, i.e. $\mathrm{wt}(\boldsymbol{x}) \in \big[(1/2-\epsilon')n, (1/2+\epsilon')n\big]$. Since $\epsilon' = \lfloor \epsilon n \rfloor / n \leqslant \epsilon n / n = \epsilon$, we have $\big[(1/2-\epsilon')n, (1/2+\epsilon')n\big] \subseteq \big[(1/2-\epsilon)n, (1/2+\epsilon)n\big]$. Therefore, $\boldsymbol{x}$ is also ϵ-balanced.

■

According to Proposition 1, it suffices to assume that pn and ϵn are integers. Clearly, when pn (or ϵn, respectively) is not an integer, by setting $p' = \lfloor np \rfloor / n$ (or setting $\epsilon' = \lfloor \epsilon n \rfloor / n$, respectively), we then obtain $p'n$ (or $\epsilon' n$) is an integer. Throughout this work, for simplicity, we suppose that n is even, and pn and ϵn are integers.

Given n, m, p, ϵ, where n is even, $\epsilon \in [0, 1/2], p \in [0,1]$, $m \leqslant n$, $pn, \epsilon n$ are integers, we set
$$\mathrm{B_{RC}}(n;p) \triangleq \Big\{ A \in \boldsymbol{A}_n : A_i, A^i \text{ are } p\text{-bounded for } 1 \leqslant i \leqslant n \Big\},$$
$$\mathrm{Bal_{RC}}(n;\epsilon) \triangleq \Big\{ A \in \boldsymbol{A}_n : A_i, A^i \text{ are } \epsilon\text{-balanced for } 1 \leqslant i \leqslant n \Big\},$$
and
$$\mathrm{B_{SW}}(n,m;p) \triangleq \Big\{ A \in \boldsymbol{A}_n : \text{every window } W \text{ of } A \text{ of size } m \times m \text{ are } p\text{-bounded} \Big\},$$
$$\mathrm{Bal_{SW}}(n,m;\epsilon) \triangleq \Big\{ A \in \boldsymbol{A}_n : \text{every window } W \text{ of } A \text{ of size } m \times m \text{ are } \epsilon\text{-balanced} \Big\}.$$
In this work, we are interested in the problem of designing efficient coding methods that encode (decode) binary data to (from) $\mathrm{B_{RC}}(n;p)$, $\mathrm{Bal_{RC}}(n;\epsilon)$, $\mathrm{B_{SW}}(n,m;p)$ and $\mathrm{Bal_{SW}}(n,m;\epsilon)$.

Definition 3: The map $\textsc{Enc} : \{0,1\}^k \to \{0,1\}^{n \times n}$ is a *2D p-bounded RC encoder* if $\textsc{Enc}(\boldsymbol{x}) \in \mathrm{B_{RC}}(n;p)$ for all $\boldsymbol{x} \in \{0,1\}^k$ and there exists a *decoder* map $\textsc{Dec}: \{0,1\}^{n \times n} \to \{0,1\}^k$ such that $\textsc{Dec} \circ \textsc{Enc}(\boldsymbol{x}) = \boldsymbol{x}$. The *coding rate of the encoder* is measured by k/n^2 and the *redundancy of the encoder* is measured by the value $n^2 - k$ (bits). The *2D ϵ-balanced RC encoder*, *2D p-bounded m-SW encoder*, and *2D ϵ-balanced m-SW encoder* are similarly defined.

Our design objectives include low redundancy (equivalently, high code rate) and low complexity of the encoding/decoding algorithms. In this work, we show that for sufficiently large n, there exist efficient encoders/decoders for $\mathrm{B_{RC}}(n;p)$, $\mathrm{Bal_{RC}}(n;\epsilon)$, $\mathrm{B_{SW}}(n,m;p)$ and $\mathrm{Bal_{SW}}(n,m;\epsilon)$, that incur at most one redundant bit.

B. Literature Works on 2D p-Bounded RC Constrained Codes $\mathrm{B_{RC}}(n;p)$

We briefly describe the works by Ordentlich and Roth in [11] and [17] that provided encoding/decoding algorithms for $\mathrm{B_{RC}}(n;p)$.

- For $p = 1/2$, i.e. the weight of every row and every column is at most $n/2$, Ordentlich and Roth [11] presented two low complexity coding methods. The first method is based on flipping rows and columns of an arbitrary binary array (i.e. using the complement of rows and columns) until the weight-constraint is satisfied in all rows and columns while the second method is based on efficient construction of *antipodal matching*. Both codes have roughly $2n$ redundant bits. A lower bound on the optimal redundancy was shown to be $\lambda n + o(n)$ for a constant $\lambda \approx 1.42515$ in [21]. Note that these two methods can be used to construct $\mathrm{B_{RC}}(n;p)$ for arbitrary $p > 1/2$, in which $\mathrm{B_{RC}}(n;1/2) \subset \mathrm{B_{RC}}(n;p)$.
- For $p < 1/2$, one may follow the coding method in [17], based on enumeration coding, that ensure the weight in every row and every column to be precisely pn. The redundancy of the proposed encoder was at most $2n\mu(n,p)$, where $\mu(n,p)$ is the least redundancy required to encode one-dimensional binary codewords of length n such that the weight is pn. If we set $\mathrm{Q}(n,p) = \Big\{ \boldsymbol{x} \in \{0,1\}^n : \mathrm{wt}(\boldsymbol{x}) = pn \Big\}$, then we have $\mu(n,p) = nH(p) - \log|\mathrm{Q}(n,p)|$ where $H(p) = -p\log p - (1-p)\log(1-p)$. It is easy to verify that $\mu(n,p) = \Theta(n)$ for all $p < 1/2$. In addition, Ordentlich and Roth [17] also provided a lower bound on the optimal redundancy, which is at least $2n\mu(n,p) + O(n + \log n)$ bits.

In this work, we first propose efficient coding methods for $\mathrm{B_{RC}}(n;p)$ when $p > 1/2$ or $p < 1/2$. Particularly, when $p < 1/2$, the redundancy can be reduced to be at most

$n\mu(n,p)+O(n+\log n)$ bits. When $p>1/2$, the redundancy can be reduced significantly to be at most $\Theta(n)$ bits and then to only one bit. We then extend the results to design efficient encoders for 2D p-bounded m-SW constrained codes $\mathrm{B_{SW}}(n,m;p)$ when $m=n-o(n)$ and $p\geq 1/2$.

We review below the antipodal matching (defined in [11]) as it will be used in one of our proposed coding methods.

Definition 4 (Ordentlich and Roth [11]): An antipodal matching ϕ is a mapping from $\{0,1\}^n$ to itself with the following properties holding for every $\boldsymbol{x}\in\{0,1\}^n$:

1) $\mathrm{wt}(\phi(\boldsymbol{x}))=n-\mathrm{w}(\boldsymbol{x})$.
2) If $\mathrm{wt}(\boldsymbol{x})>n/2$ then $\phi(\boldsymbol{x})$ has all its 1's in positions where $\boldsymbol{x}$ has 1's. In other words, suppose $\boldsymbol{x}=x_1x_2\ldots x_n$ and $\boldsymbol{y}=\phi(\boldsymbol{x})=y_1y_2\ldots y_n$, then $y_i=1$ implies $x_i=1$ for $1\leqslant i\leqslant n$.
3) $\phi(\phi(\boldsymbol{x}))=\boldsymbol{x}$.

Ordentlich and Roth [11] presented an efficient construction of antipodal matchings ϕ for all n. In fact, such an antipodal matching can be decomposed into a collection of bijective mappings $\phi=\cup_{i=0}^{n}\phi_i$, where

$$\phi_i:\Big\{\boldsymbol{x}\in\{0,1\}^n:\mathrm{wt}(\boldsymbol{x})=i\Big\}\to\Big\{\boldsymbol{x}\in\{0,1\}^n:\mathrm{wt}(\boldsymbol{x})=n-i\Big\},$$

and ϕ_i can be constructed in linear-time for all n,i as in [11].

C. Literature Works on 2D ϵ-Balanced RC Constrained Codes $\mathrm{Bal_{RC}}(n;\epsilon)$

Over 1D codes, to encode binary balanced codes, we have the celebrated *Knuth's balancing technique* [19]. Knuth's balancing technique is a linear-time algorithm that maps a binary message $\boldsymbol{x}$ to a balanced word $\boldsymbol{z}$ of the same length by flipping the first t bits of $\boldsymbol{x}$. The crucial observation demonstrated by Knuth is that such an index t always exists and t is commonly referred to as a *balancing index* of $\boldsymbol{x}$. To represent such a balancing index, Knuth appends $\boldsymbol{z}$ with a short balanced suffix of length $\lceil\log n\rceil$, which is also the redundancy of the encoding algorithm. Formally, we have the following theorem.

Theorem 1 (Knuth [19] for $\epsilon=0$): For arbitrary binary sequence $\boldsymbol{x}\in\{0,1\}^n$, there exists the index t, where $1\leq t\leq n$, called a balancing index of $\boldsymbol{x}$ such that the sequence $\boldsymbol{y}$ obtained by flipping the first t bits in $\boldsymbol{x}$, denoted by $\mathrm{Flip}_t(\boldsymbol{x})$, is balanced. There exists a pair of linear-time algorithms $\textsc{Enc}^K:\{0,1\}^k\to\{0,1\}^n$ and $\textsc{Dec}^K:\{0,1\}^n\to\{0,1\}^k$, where $k\approx n-\lceil\log n\rceil$, such that the following holds. If $\textsc{Enc}^K(\boldsymbol{x})$ is balanced and $\textsc{Dec}^K\circ\textsc{Enc}^K(\boldsymbol{x})=\boldsymbol{x}$ for all $\boldsymbol{x}\in\{0,1\}^k$.

Modifications of the generic scheme are discussed for constructing more efficient balanced codes [20], [22] and almost-balanced codes [23], [24], [25]. Particularly, for 1D ϵ-balanced codes, the encoding methods in [23] and [25] used only a constant redundant bits. Crucial to the improvement in redundancy from $\lceil\log n\rceil$ bit of Knuth's method for balanced codes to a constant bits for ϵ-balanced codes is the construction of *ϵ-balancing set*.

Definition 5: For n even, $\epsilon>0,\epsilon n\geqslant 1$, the index t, where $1\leq t\leq n$, is called an *ϵ-balancing index* of $\boldsymbol{x}\in\{0,1\}^n$ if $\boldsymbol{y}=\mathrm{Flip}_t(\boldsymbol{x})$, is *$\epsilon$-balanced*. Suppose that $\epsilon n>1$, let the *ϵ-balancing set* $\mathrm{S}_{\epsilon,n}\subset\{0,1,2,\ldots,n\}$ be the set of indices, given by $\mathrm{S}_{\epsilon,n}=\{0,n\}\cup\{2\lfloor\epsilon n\rfloor,4\lfloor\epsilon n\rfloor,6\lfloor\epsilon n\rfloor,\ldots\}$. The size of $\mathrm{S}_{\epsilon,n}$ is at most $\lfloor 1/2\epsilon\rfloor+1$.

Theorem 2 (Nguyen et al. [25]): Let n be even, and $\epsilon>0,\epsilon n\geqslant 1$. For arbitrary binary sequence $\boldsymbol{x}\in\{0,1\}^n$, there exists an index t in the set $\mathrm{S}_{\epsilon,n}$, such that t is an ϵ-balancing index of $\boldsymbol{x}$. There exists a pair of linear-time algorithms $\textsc{Enc}^\epsilon:\{0,1\}^k\to\{0,1\}^n$ and $\textsc{Dec}^\epsilon:\{0,1\}^n\to\{0,1\}^k$, where $k=n-2\log(\lfloor 1/2\epsilon\rfloor+1)$, such that the following holds. If $\textsc{Enc}^\epsilon(\boldsymbol{x})$ is ϵ-balanced and $\textsc{Dec}^\epsilon\circ\textsc{Enc}^\epsilon(\boldsymbol{x})=\boldsymbol{x}$ for all $\boldsymbol{x}\in\{0,1\}^k$. The redundancy of the encoder is $2\log(\lfloor 1/2\epsilon\rfloor+1)=\Theta(1)$.

Since an ϵ-balancing index can be found in $\mathrm{S}_{\epsilon,n}$ of constant size, to encode such an ϵ-balancing index t, only a constant redundant bits is needed.

Example 2: Consider $n=10,\epsilon=0.1$, we then have $\mathrm{S}_{\epsilon,n}=\{0,2,4,6,8\}$. Let $\boldsymbol{x}=0000000000$. Observe that $f_4(\boldsymbol{x})=1111000000$, and $f_6(\boldsymbol{x})=1111110000$ are ϵ-balanced. Hence, $t=4,6$ are ϵ-balancing indices of $\boldsymbol{x}$. In general, there might be more than one ϵ-balancing index.

For 2D ϵ-balanced RC codes $\mathrm{Bal_{RC}}(n;\epsilon)$ and $\epsilon=0$, Talyansky et al. [3] presented an efficient encoding method for $n\times n$ array, where each row and each column is balanced. The method uses $2n\log n+\Theta(n\log\log n)$ redundant bits and includes three phases.

- In phase I, all rows in the array are encoded to be balanced via the Knuth's balancing technique, which uses roughly $\log n$ redundant bits for each row.
- In phase II, the array is then divided into two subarrays of equal size, and the problem is reduced to balancing two subarrays from a given balanced array. The process is repeated until each subarray is a single column. For each subproblem, to balance two subarrays, the authors simply *swap* the elements between two subarrays until both subarrays are balanced. The number of swapped elements, called *index*, will be encoded and decoded in phase III in order to recover the information.
- In phase III, the author presented a method to encode/decode all the indices used in phase II recursively. The authors also proved that the redundancy in this phase is at most $n\log n+O(\log\log n)$ bits.

To further reduce the redundancy, instead of using one of the algorithms in [19] and [20], Talyansky et al. balanced the rows with the (more computationally complex) enumerative coding technique. Consequently, the redundancy can be reduced to $1.5n\log n+\Theta(n\log\log n)$ redundant bits.

In this work, we first extend the study in the literature works to design 2D ϵ-balanced RC codes $\mathrm{Bal_{RC}}(n;\epsilon)$ for arbitrary $\epsilon>0$. Particularly, for $n>0,\epsilon\in(0,1/2)$, we design efficient coding methods that encode (decode) binary data to (from) $\mathrm{Bal_{RC}}(n;\epsilon)$. Clearly, since $\mathrm{Bal_{RC}}(n;0)\subset\mathrm{Bal_{RC}}(n;\epsilon)$ for all $\epsilon>0$, one may use the constructions of Talyansky et al. [3] that use at most $1.5n\log n+\Theta(n\log\log n)$ redundant bits. Throughout this work, we show that the redundancy can be reduced significantly to be at most $n\log n+o(n\log n)$ redundant bits. Furthermore, for $\epsilon>0$ and for sufficiently large n, we show that there exist linear-time encoding

TABLE I

A Summary of the Encoders and Decoders Proposed by This Work. The Redundancy Is Computed for Array Codewords of Length $n \times n$, n Is Even, $p, \epsilon > 0$

Encoder / Decoder	Description	Redundancy	Input Requirement
$\text{Enc}^1_{\text{B}_{\text{RC}}(n;p)}$, $\text{Dec}^1_{\text{B}_{\text{RC}}(n;p)}$	Encoder and decoder for $\text{B}_{\text{RC}}(n;p)$: the weight of every row and every column is at most pn when $p < 1/2$ using the divide and conquer method	$n\lambda(n,p) + O(n \log n)$ bits	$n > \lceil 1/p \rceil (1 + \log n)$
$\text{Enc}^2_{\text{B}_{\text{RC}}(n;p)}$, $\text{Dec}^2_{\text{B}_{\text{RC}}(n;p)}$	Encoder and decoder for $\text{B}_{\text{RC}}(n;p)$: the weight of every row and every column is at most pn when $p > 1/2$ using the sequence replacement technique	$(n+3)$ bits	$n \geqslant 1/c^2 \ln(n^2 - n - 2)$ where $c = p - 1/2$
$\text{Enc}^1_{\text{Bal}_{\text{RC}}(n;\epsilon)}$, $\text{Dec}^1_{\text{Bal}_{\text{RC}}(n;\epsilon)}$	Encoder and decoder for $\text{Bal}_{\text{RC}}(n;\epsilon)$: the weight of every row and every column is within $[(1/2 - \epsilon)n, (1/2 + \epsilon)n]$ using the divide and conquer method	$n \log n + o(n \log n)$ bits	$\epsilon > 0, \epsilon n \geqslant 1$
$\text{Enc}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}$, $\text{Dec}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}$	Encoder and decoder for $\text{Bal}_{\text{RC}}(n;\epsilon)$: the weight of every row and every column is within $[(1/2 - \epsilon)n, (1/2 + \epsilon)n]$ using the sequence replacement technique	1 bit	$n^2 \geqslant 8/\epsilon^2 \ln n, n\epsilon \geqslant 2$
$\text{Enc}_{\text{B}_{\text{SW}}(n,m;p)}$, $\text{Dec}_{\text{B}_{\text{SW}}(n,m;p)}$	Encoder and decoder for $\text{B}_{\text{SW}}(n,m;p)$: the weight of every window of size $m \times m$ is at most pm^2 where $m = n - k$, and $p \geqslant 1/2$ using the antipodal matching	$2(k+1)^2$ bits	$k = o(n)$
$\text{Enc}_{\text{Bal}_{\text{SW}}(n,m;\epsilon)}$, $\text{Dec}_{\text{Bal}_{\text{SW}}(n,m;\epsilon)}$	Encoder and decoder for $\text{Bal}_{\text{S}}(n,m;\epsilon)$: the weight of every window of size $m \times m$ is within $[(1/2-\epsilon)m^2, (1/2+\epsilon)m^2]$ using the sequence replacement technique	1 bit	$2/\epsilon^2 \ln n \leqslant m \leqslant n$

(and decoding respectively) algorithms for $\text{Bal}_{\text{RC}}(n;\epsilon)$ that use only one redundant bit.

D. Paper Organisation and Our Contribution

In this work, we present efficient encoding/decoding methods for 2D RC constrained codes and 2D SW constrained codes for arbitrary $p \in [0,1]$ and $\epsilon \in [0, 1/2]$. Method A is based on the divide and conquer algorithm and a modification of the Knuth's balancing technique while method B is based on the sequence replacement technique. We present a summary of the encoders and decoders proposed by this work in Table I.

(A) In Section III, we uses method A to encode $\text{B}_{\text{RC}}(n;p)$ when $p \leqslant 1/2$ with at most $n\mu(n,p) + O(n + \log n)$ redundant bits (compared to $2n\mu(n,p) + O(n + \log n)$ if using the method in [17]), and to encode $\text{Bal}_{\text{RC}}(n;\epsilon)$ for arbitrary $\epsilon \in (0, 1/2)$ with at most $n \log n + o(n \log n)$ redundant bits (compared to $1.5n \log n + \Theta(n \log \log n)$ if using the method [3]).

(B) In Section IV, we first uses method B to encode $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$ with at most $n+3$ redundant bits. We then show that for sufficiently large n, we can encode $\text{Bal}_{\text{RC}}(n;\epsilon)$ or encode $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$ with only one redundant bit.

(C) In Section V, we propose encoding/decoding methods for 2D SW constrained codes $\text{B}_{\text{SW}}(n,m;p)$ and $\text{Bal}_{\text{SW}}(n,m;\epsilon)$ when $m = n - o(n)$ and $p \geq 1/2$.

III. Efficient Encoders/Decoders for $\text{B}_{\text{RC}}(n;p)$ and $\text{Bal}_{\text{RC}}(n;\epsilon)$ via The Divide and Conquer Algorithm

We first define the swapping function of two binary sequences.

Definition 6: Given $\boldsymbol{y} = y_1y_2\ldots y_n, \boldsymbol{z} = z_1z_2\ldots z_n$. For $1 \leqslant t \leqslant n$, we use $\text{Swap}_t(\boldsymbol{y},\boldsymbol{z})$, $\text{Swap}_t(\boldsymbol{z},\boldsymbol{y})$ to denote the sequences obtained by swapping the first t bits of $\boldsymbol{y}$ and $\boldsymbol{z}$, i.e.

$$\text{Swap}_t(\boldsymbol{y},\boldsymbol{z}) = z_1z_2\ldots z_t y_{t+1}y_{t+2}\ldots y_n, \text{ and}$$
$$\text{Swap}_t(\boldsymbol{z},\boldsymbol{y}) = y_1y_2\ldots y_t z_{t+1}z_{t+2}\ldots z_n.$$

A key ingredient of the encoding method in [3] is the following lemma.

Lemma 1 (Swapping Lemma): Given n is even, $\boldsymbol{x} = \boldsymbol{yz} \in \{0,1\}^{2n}$, $\boldsymbol{x}$ is balanced, where $\boldsymbol{y} = y_1y_2\ldots y_n, \boldsymbol{z} = z_1z_2\ldots z_n$. There exists an index t, $1 \leqslant t \leqslant n$ such that both $\text{Swap}_t(\boldsymbol{y},\boldsymbol{z})$ and $\text{Swap}_t(\boldsymbol{z},\boldsymbol{y})$ are balanced.

Lemma 1 states that there must be an index t, referred as a *swapping index* of $\boldsymbol{y}$ and $\boldsymbol{z}$, such that after swapping their first t bits, the resulting sequences are both balanced. Since such an index t belongs to $\{1, 2, \ldots n\}$, in order to recover the original sequences $\boldsymbol{y}, \boldsymbol{z}$ from the output sequences $\text{Swap}_t(\boldsymbol{y},\boldsymbol{z})$ and $\text{Swap}_t(\boldsymbol{z},\boldsymbol{y})$, a redundancy of $\log n$ (bits) is needed. Therefore, to encode a binary array of size $n \times n$, the method in [3] used a total of $n \log n + O(\log \log n)$ redundant bits to store all swapping indices of all pairs of subarrays. It is easy to verify that Lemma 1 also works for p-bounded constraint and ϵ-balanced constraint for all $p, \epsilon > 0$. We have the following corollary.

Corollary 1: Given $n > 0$, $\boldsymbol{x} = \boldsymbol{yz} \in \{0,1\}^{2n}$, where $\boldsymbol{y} = y_1y_2\ldots y_n, \boldsymbol{z} = z_1z_2\ldots z_n$.

(i) Given $p \in [0,1]$, np is integer, if $\boldsymbol{x}$ is p-bounded then there exists an index t, $1 \leqslant t \leqslant n$ such that both $\text{Swap}_t(\boldsymbol{y},\boldsymbol{z})$ and $\text{Swap}_t(\boldsymbol{z},\boldsymbol{y})$ are p-bounded.

(ii) Given $\epsilon \in [0, 1/2]$, ϵn is integer, n is even, if $\boldsymbol{x}$ is ϵ-balanced then there exists an index t, $1 \leqslant t \leqslant n$ such that both $\text{Swap}_t(\boldsymbol{y},\boldsymbol{z})$ and $\text{Swap}_t(\boldsymbol{z},\boldsymbol{y})$ are ϵ-balanced.

Proof: W.l.o.g, we suppose that $\text{wt}(\boldsymbol{y}) \geqslant \text{wt}(\boldsymbol{z})$.

(i) We first show that if $\boldsymbol{x}$ is p-bounded then there exists an index t_1, $1 \leqslant t_1 \leqslant n$ such that both $\text{Swap}_{t_1}(\boldsymbol{y},\boldsymbol{z})$ and $\text{Swap}_{t_1}(\boldsymbol{z},\boldsymbol{y})$ are p-bounded. Clearly, if both $\boldsymbol{y}$ and $\boldsymbol{z}$ are p-bounded then we can select $t_1 = n$ since $\text{Swap}_n(\boldsymbol{y},\boldsymbol{z}) = \boldsymbol{z}$ and $\text{Swap}_n(\boldsymbol{z},\boldsymbol{y}) = \boldsymbol{y}$. On the other hand, if one of them is not p-bounded, suppose that we have $\mathrm{wt}(\boldsymbol{y}) > pn$ while $\mathrm{wt}(\boldsymbol{z}) < pn$.

- We then swap the bits in $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively in the order from left to right. Observe that each swap increases or decreases the weight of $\boldsymbol{y}$ (or $\boldsymbol{z}$) by one, and when all the bits in $\boldsymbol{y}$ and $\boldsymbol{z}$ are swapped, i.e. $t = n$, we have $\mathrm{wt}\big(\mathrm{Swap}_n(\boldsymbol{y}, \boldsymbol{z})\big) = \mathrm{wt}(\boldsymbol{z}) < pn$. Therefore, there must be an index t_1, $1 \leqslant t_1 \leqslant n$ such that after swapping t_1 bits, we have $\mathrm{wt}\big(\mathrm{Swap}_{t_1}(\boldsymbol{y}, \boldsymbol{z})\big) = pn$. It remains to show that, for such an index t_1, we have $\mathrm{wt}\big(\mathrm{Swap}_{t_1}(\boldsymbol{z}, \boldsymbol{y})\big) \leqslant pn$.
- Observe that $\mathrm{wt}\big(\mathrm{Swap}_t(\boldsymbol{y}, \boldsymbol{z})\big) + \mathrm{wt}\big(\mathrm{Swap}_t(\boldsymbol{z}, \boldsymbol{y})\big) = \mathrm{wt}(\boldsymbol{y}) + \mathrm{wt}(\boldsymbol{z}) = \mathrm{wt}(\boldsymbol{x}) \leqslant p(2n) = 2pn$ for all $1 \leqslant t \leqslant n$. Consequently, if $\mathrm{wt}\big(\mathrm{Swap}_{t_1}(\boldsymbol{y}, \boldsymbol{z})\big) = pn$, we then have $\mathrm{wt}\big(\mathrm{Swap}_{t_1}(\boldsymbol{z}, \boldsymbol{y})\big) \leqslant pn$. In conclusion, for such an index t_1, both $\mathrm{Swap}_{t_1}(\boldsymbol{y}, \boldsymbol{z})$ and $\mathrm{Swap}_{t_1}(\boldsymbol{z}, \boldsymbol{y})$ are p-bounded.

In conclusion, there always exists an index t_1, $1 \leqslant t_1 \leqslant n$ such that both $\mathrm{Swap}_{t_1}(\boldsymbol{y}, \boldsymbol{z})$ and $\mathrm{Swap}_{t_1}(\boldsymbol{z}, \boldsymbol{y})$ are p-bounded.

(ii) We now show that if $\boldsymbol{x}$ is ϵ-balanced then there exists an index t_2, $1 \leqslant t_2 \leqslant n$ such that both $\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})$ and $\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})$ are ϵ-balanced. Since $\boldsymbol{x}$ is ϵ-balanced, i.e. $\mathrm{wt}(\boldsymbol{x}) \in [(1/2-\epsilon)2n, (1/2+\epsilon)2n]$, we then have $n \leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n + 2\epsilon n$, or $n - 2\epsilon n \leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n$. We first consider the case when $n \leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n + 2\epsilon n$.

Since $\mathrm{wt}(\boldsymbol{y}) \geqslant \mathrm{wt}(\boldsymbol{z})$, it implies that $\mathrm{wt}(\boldsymbol{y}) \geqslant n/2$ and $\mathrm{wt}(\boldsymbol{z}) \leqslant n/2 + \epsilon n$. Clearly, if both $\boldsymbol{y}$ and $\boldsymbol{z}$ are ϵ-balanced then we can select $t_2 = n$ since $\mathrm{Swap}_n(\boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{z}$ and $\mathrm{Swap}_n(\boldsymbol{z}, \boldsymbol{y}) = \boldsymbol{y}$. On the other hand, suppose that at least one of them is not ϵ-balanced.

- If $\boldsymbol{y}$ is not ϵ-balanced, we conclude that $\mathrm{wt}(\boldsymbol{y}) > (1/2+\epsilon)n = n/2 + \epsilon n$. We then have $\mathrm{wt}(\boldsymbol{z}) < n/2 + \epsilon n$ since $\mathrm{wt}(\boldsymbol{x}) \leqslant n + 2\epsilon n$. We then swap the bits in $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively in the order from left to right. Observe that each swap increases or decreases the weight of $\boldsymbol{y}$ (or $\boldsymbol{z}$) by one, and when $t = n$, we have $\mathrm{wt}\big(\mathrm{Swap}_n(\boldsymbol{y}, \boldsymbol{z})\big) = \mathrm{wt}(\boldsymbol{z}) < n/2 + \epsilon n$. Therefore, there must be an index t_2, $1 \leqslant t_2 \leqslant n$ such that after swapping t_2 bits, we have $\mathrm{wt}\big(\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})\big) = n/2 + \epsilon n$. In other words, $\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})$ is ϵ-balanced. It remains to show that $\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})$ is ϵ-balanced. Observe that $\mathrm{wt}\big(\mathrm{Swap}_t(\boldsymbol{y}, \boldsymbol{z})\big) + \mathrm{wt}\big(\mathrm{Swap}_t(\boldsymbol{z}, \boldsymbol{y})\big) = \mathrm{wt}(\boldsymbol{y}) + \mathrm{wt}(\boldsymbol{z}) = \mathrm{wt}(\boldsymbol{x}) \in [n, n+2\epsilon n]$ for all $1 \leqslant t \leqslant n$. Therefore, when $t = t_2$, if $\mathrm{wt}\big(\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})\big) = n/2 + \epsilon n$, we then have
$$\mathrm{wt}\big(\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})\big) \geqslant n - (n/2 + \epsilon n) = n/2 - \epsilon n, \text{ and}$$
$$\mathrm{wt}\big(\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})\big) \leqslant n + 2\epsilon n - (n/2 + \epsilon n) = n/2 + \epsilon n.$$
Thus, $\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})$ is ϵ-balanced.
- On the other hand, if $\boldsymbol{z}$ is not ϵ-balanced, we have $\mathrm{wt}(\boldsymbol{z}) \notin [n/2 - \epsilon n, n/2 + \epsilon n]$. Since $\mathrm{wt}(\boldsymbol{z}) \leqslant n/2 + \epsilon n$, we conclude that $\mathrm{wt}(\boldsymbol{z}) < n/2 - \epsilon n$. Moreover, since $\mathrm{wt}(\boldsymbol{x}) \geqslant n$, it implies $\mathrm{wt}(\boldsymbol{y}) > n/2 + \epsilon n$, or $\boldsymbol{y}$ is also not ϵ-balanced. We have shown that there exists an index t_2 such that both $\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})$ and $\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})$ are ϵ-balanced in the earlier case.

We now consider the case when $n/2 - \epsilon n \leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n$. One can obtain similar proof to the case $n \leqslant \mathrm{wt}(\boldsymbol{x}) \leqslant n + 2\epsilon n$. Here, we use the idea of the complement sequence as follows. Observe that a binary sequence $\boldsymbol{x}$ is ϵ-balanced if and only if its complement, $\overline{\boldsymbol{x}}$ is also ϵ-balanced. We then obtain $\overline{\boldsymbol{x}} = \overline{\boldsymbol{y}}\,\overline{\boldsymbol{z}}$ where $n \leqslant \mathrm{wt}(\overline{\boldsymbol{x}}) \leqslant n + 2\epsilon n$. Here, $\overline{\boldsymbol{x}}$ is ϵ-balanced. We have shown that there exists an index t_2 such that $\mathrm{Swap}_{t_2}(\overline{\boldsymbol{y}}, \overline{\boldsymbol{z}})$ and $\mathrm{Swap}_{t_2}(\overline{\boldsymbol{z}}, \overline{\boldsymbol{y}})$ are both ϵ-balanced. Consequently, we have $\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})$ and $\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})$ are also ϵ-balanced.

In conclusion, there always exists an index t_2, $1 \leqslant t_2 \leqslant n$ such that both $\mathrm{Swap}_{t_2}(\boldsymbol{y}, \boldsymbol{z})$ and $\mathrm{Swap}_{t_2}(\boldsymbol{z}, \boldsymbol{y})$ are ϵ-balanced.

■

We now present an efficient encoding method for 2D p-bounded RC constrained codes $\mathrm{B}_{\mathrm{RC}}(n; p)$ for given $p < 1/2$ and for 2D ϵ-balanced RC constrained codes $\mathrm{Bal}_{\mathrm{RC}}(n; \epsilon)$ for $\epsilon \in (0, 1/2)$ via the divide and conquer algorithm.

A. Design of $\mathrm{B}_{\mathrm{RC}}(n; p)$ When $p < 1/2$

Recall that one may follow the coding method in [17], based on enumeration coding, that ensure the weight in every row and every column to be precisely pn. The redundancy of the proposed encoder was at most $2n\mu(n,p) + O(n + \log n)$, where $\mu(n,p)$ is the least redundancy required to encode one-dimensional binary codewords of length n such that the weight is pn (refer to Section II-B).

In this section, we adapt the divide and conquer algorithm with the enumeration coding technique. Compare to literature works in [17] and [26] that also used modifications of the enumeration coding technique, the major difference of our coding method is that the rows and columns are encoded independently, and the 2D code construction can be divided into two 1D code constructions. In other words, the complexity of our encoder mainly depends on the efficiency of enumeration coding for 1D codes. Hence, the complexity of our encoder mainly depends on the efficiency of enumeration coding for one-dimensional codes.

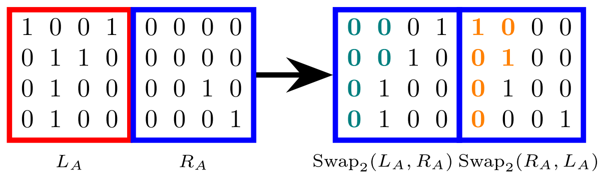
In general, a *ranking function* for a finite set S of cardinality N is a bijection $\mathrm{rank} : \mathrm{S} \to [N]$ where $[N] \triangleq \{0, 1, 2, \ldots, N-1\}$. Associated with the function rank is a unique *unranking function* $\mathrm{unrank} : [N] \to \mathrm{S}$, such that $\mathrm{rank}(s) = j$ if and only if $\mathrm{unrank}(j) = s$ for all $s \in \mathrm{S}$ and $j \in [N]$. Given $n, p > 0$, let $\mathrm{S}(n,p) \triangleq \Big\{\boldsymbol{x} \in \{0,1\}^n : \mathrm{wt}(\boldsymbol{x}) \leqslant pn\Big\}$, i.e. $\mathrm{S}(n,p)$ is the set of all one-dimensional sequences that satisfy the p-bounded constraint. One may use enumeration coding [27], [28] to construct $\mathrm{rank}_p : \mathrm{S}(n,p) \to [|\mathrm{S}(n,p)|]$ and $\mathrm{unrank}_p : [|\mathrm{S}(n,p)|] \to \mathrm{S}(n,p)$. The redundancy of this encoding algorithm is then $\lambda(n,p) = n - \log|\mathrm{S}(n,p)| \leqslant \mu(n,p)$ (bits).

We now describe briefly the main idea of the algorithm. We first consider the case when n is a power of 2, and hence, $\log n$ is an integer. Recall from Proposition 1, it suffices to assume that np is an integer.

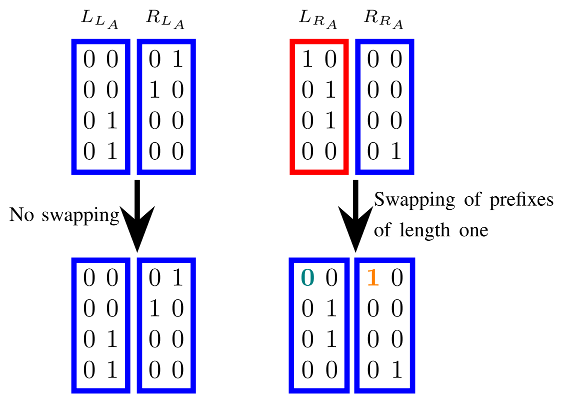
Encoder $\mathrm{ENC}^{1}_{\mathrm{B}_{\mathrm{RC}}(n;p)}$. Let c be the smallest integer such that $c \geqslant \lceil 1/p \rceil (1 + \log n)$ and cp is an integer. When $1/p$ is an integer, we can set $c = 1/p(1 + \log n)$. In general, we have $c = O(\log n)$. The binary data $\boldsymbol{x}$ is of length $N = (n - c)(n - \lambda(n,p))$, i.e. the redundancy is then $n\lambda(n,p) + cn - c\lambda(n,p) = n\lambda(n,p) + O(n \log n)$ bits. The encoding method includes three phases.

- *Phase I: Enforcing rows to be p-bounded.* The encoder encodes the information of length N into an array A of size $(n-c) \times n$ where every row is p-bounded, using the enumeration coding technique. Particularly, the information is encoded into $\mathrm{S}(n,p)$ with the redundancy at most $\lambda(n,p)$ bits for each row. Therefore, the redundancy used in Phase I is at most $(n-c)\lambda(n,p) < n\lambda(n,p)$ bits.
- *Phase II: Enforcing columns to be p-bounded.* The encoder ensures that every column of A is p-bounded. From Phase I, every row of A is p-bounded, and hence, array A is p-bounded. Suppose that at some encoding step i, we have an array S of size $(n-c) \times n_0$, which is already p-bounded for some $n_0 \leqslant n$, n_0 is a power of 2 (initially, $n_0 \equiv n$).
  Dividing S. The encoder divides S into two subarrays of size $(n-c) \times (n_0/2)$, denoted by L_S and R_S, where L_S consists of the first $n_0/2$ columns of S and R_S consists of the last $n_0/2$ columns of S. The encoder proceeds to ensure that L_S and R_S are both p-bounded as follows.
  SWAP(L_S, R_S). The encoder follows Corollary 1 to find a swapping index so that L_S and R_S are both p-bounded. To represent such a swapping index t, we need at most $\log((n-c)n_0/2)$ redundant bits.
  After both subarrays L_S and R_S are p-bounded, the encoder proceeds to divide each of them into two subarrays and repeats the process to ensure that the newly created subarrays are also p-bounded. This process ends when all subarrays of size $(n-c) \times 1$ are p-bounded. We illustrate the idea of the swapping procedure in Figure 1.
  Let $\mathrm{Re}(n)$ be the sequence obtained by concatenating all binary representations of all swapping indices. Then
$$|\mathrm{Re}(n)| \leqslant \sum_{k=2^j\, 2\leqslant k\leqslant n} (n/k)\log((n-c)k/2)$$
$$= (n-1)(1+\log(n-c)) - \log n$$
$$< n(1+\log n) \text{ (bits).}$$
- *Phase III: Encoding the swapping indices.* The encoder encodes $\mathrm{Re}(n)$ into an array B of size $c \times n$ such that its every row and every column is p-bounded. At the end of Phase III, the encoder outputs the concatenation of A and B, which is an array of size $n \times n$. In contrast to the encoder in [3], which proceeds to repeat the encoding procedure to encode $\mathrm{Re}(n)$, we show that $\mathrm{Re}(n)$ can be encoded/decoded efficiently without repeating the encoding procedure. Recall that $c \geqslant \lceil 1/p \rceil (1 + \log n)$ and np is integer. Suppose that $\mathrm{Re}(n) = x_1 x_2 \ldots x_{n(1+\log n)}$, we fill the bits in $\mathrm{Re}(n)$ into the array B column by column as follows: the first column $B^1 = (x_1 0^{\lceil \frac{1}{p} \rceil - 1})(x_2 0^{\lceil \frac{1}{p} \rceil - 1}) \ldots (x_{1+\log n} 0^{\lceil \frac{1}{p} \rceil - 1})$, and for $2 \leqslant i \leqslant n$ the ith column is $B^i = (0^{j-1} x_{(1+\log n)(i-1)+1} 0^{\lceil \frac{1}{p} \rceil - j}) \ldots (0^{j-1} x_{(1+\log n)i} 0^{\lceil \frac{1}{p} \rceil - j})$, where $j \equiv i \pmod{\lceil 1/p \rceil}$. It is easy to verify that every row and every column of B is p-bounded and $\mathrm{Re}(n)$ can be decoded uniquely from B.

(a) We divide A into two subarrays L_A and R_A. In this step, only L_A is not p-bounded. To obtain two p-bounded subarrays, we swap their prefixes of length six. In other words, we set L_A and R_A to be $\mathrm{Swap}_2(L_A, R_A)$ and $\mathrm{Swap}_2(R_A, L_A)$, respectively.



(b) We divide L_A into two subarrays L_{L_A} and R_{L_A}. Here, both subarrays are p-bounded and there is no swapping of prefixes. When we divide R_A into two subarrays L_{R_A} and R_{R_A}, we observe that L_{R_A} is not p-bounded. We proceed as before.



(c) Finally, we divide each of the four subarrays into two, resulting in the $n = 8$ columns. The final output is as follows.

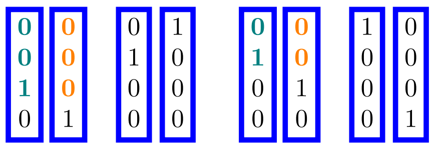


Fig. 1. Example for $n = 8, p = 1/4$. The current subarray A is of size 8×4. The subarrays, highlighted in red, are not p-bounded while those, highlighted in blue, are p-bounded.

In conclusion, the total redundancy for encoding an $n \times n$ array is bounded above by:
$$(n-c)\lambda(n,p) + (1/p)n(1+\log n) < n\lambda(n,p) + O(n\log n)$$
$$< n\mu(n,p) + O(n \log n) \text{ bits.}$$

For completeness, we present the corresponding decoding algorithm as follows.

Decoder, $\mathrm{DEC}^1_{\mathrm{B_{RC}}(n;p)}$.

INPUT: $A \in \mathrm{B_{RC}}(n;p)$ of size $n \times n$
OUTPUT: $\boldsymbol{x} \triangleq \mathrm{DEC}_{\mathrm{B}^1_{\mathrm{RC}}(n;p)}(A) \in \{0,1\}^N$, where $N = (n - c)(n - \lambda(n,p))$

(I) Let B be the subarray obtained by the last c rows of A and C be the subarray obtained by the first $(n-c)$ rows of A

(II) Decode $\mathrm{Re}(n)$ from B

(III) Do the reverse swapping process in C according to $\mathrm{Re}(n)$, C is an array of size $(n-c)\times n$. Let $\boldsymbol{y}_i$ be the ith row of C, we then have $\boldsymbol{y}_i \in \mathbf{S}(n,p)$ for all $1 \leqslant i \leqslant n-c$

(IV) For $1 \leqslant i \leqslant n-c$, let $\boldsymbol{z}_i$ be the binary sequence of length $(n-\lambda(n,p))$ representing $\mathrm{rank}(\boldsymbol{y}_i)$

(V) Output $\boldsymbol{x} \triangleq \boldsymbol{z}_1\boldsymbol{z}_2\ldots\boldsymbol{z}_{n-c} \in \{0,1\}^{(n-c)(n-\lambda(n,p))}$

Finally, we point out the changes that need to be made in Phase II when n is not a power of 2. Suppose that we have an array S of size $(n-c)\times n_0$, which is already p-bounded for some $n_0 \leqslant n$ (initially, $n_0 \equiv n$), i.e. $\mathrm{wt}(S) \leqslant p(n-c)n_0$. Here, the value of n_0 can be odd, in which case we can define L_S to consist of the first $(n_0+1)/2$ columns of S, whereas R_S consists of the remaining $(n_0-1)/2$ columns of S. Observe that $(n-c)p$ and np are integers. Clearly, if L_S and R_S are both p-bounded, then no swap is needed. Suppose, on the other hand, one of them is not p-bounded.

- Suppose that L_S is not p-bounded. In other words, we have $\mathrm{wt}(L_S) > p(n-c)(n_0+1)/2$. We then have R_S is p-bounded and $\mathrm{wt}(R_S) < p(n-c)(n_0-1)/2$. Let L'_S be the array obtained by removing the column with the minimum weight from L_S. Suppose that we remove the column j_1 for some $1 \leqslant j_1 \leqslant (n_0+1)/2$. We observe that L'_S is not p-bounded, or $\mathrm{wt}(L'_S) > p(n-c)(n_0-1)/2$. We then swap the bits from L'_S and R_S.
When $t=0$, we have
$$\mathrm{wt}(\mathrm{Swap}_0(R_S, L'_S)) = \mathrm{wt}(R_S) < \frac{p(n-c)(n_0-1)}{2}.$$
When all bits are swapped, $t=(n-c)(n_0-1)/2$,
$$\mathrm{wt}(\mathrm{Swap}_{\frac{(n-c)(n_0-1)}{2}}(R_S, L'_S)) = \mathrm{wt}(L'_S) > \frac{p(n-c)(n_0-1)}{2}$$
Therefore, there exists t, $1 \leqslant t < (n-c)(n_0-1)/2$, such that $\mathrm{wt}(\mathrm{Swap}_t(R_S, L'_S)) = p(n-c)(n_0-1)/2$. For such an index t, we now show that the subarray, consisting of $\mathrm{Swap}_t(L'_S, R_S)$ and the j_1th column, is also p-bounded. Clearly, since the total weight is unchanged for each swap, we have $\mathrm{wt}(S) \leqslant p(n-c)n_0$. Hence, if $\mathrm{wt}(\mathrm{Swap}_t(R_S, L'_S)) = p(n-c)(n_0-1)/2$ then the weight of the remaining subarray is at most $p(n-c)(n_0+1)/2$.
- Suppose that R_S is not p-bounded. In other words, $\mathrm{wt}(R_S) > p(n-c)(n_0-1)/2$. We then have L_S is p-bounded. Let L'_S be the array obtained by removing the column with the maximum weight from L_S. Suppose that we remove the column j_2, where $1 \leqslant j_2 \leqslant (n_0+1)/2$. We observe that L'_S is still p-bounded, or $\mathrm{wt}(L'_S) < p(n-c)(n_0-1)/2$. Similarly, we then swap the bits from L'_S and R_S. There must be an index t, where $1 \leqslant t < (n-c)(n_0-1)/2$, such that after swapping t bits, we obtain $\mathrm{wt}(\mathrm{Swap}_t(R_S, L'_S)) = p(n-c)(n_0-1)/2$, and the subarray, consisting of $\mathrm{Swap}_t(L'_S, R_S)$ and the j_2th column, is also p-bounded.

In conclusion, after the swapping procedure, we obtain two subarrays which are both p-bounded. The encoder adds redundancy to record the swapping index t and the index of the excluded column (that incurs $\log((n_0+1)/2)$ bits). The additional redundancy to record the indices of all excluded columns is bounded above by $\sum_{\substack{k=n/2^i \\ 0\leqslant i \leqslant \lceil \log n \rceil}} (n/k)\log\lceil k/2 \rceil = O(n)$ (bits).

B. Design of $\mathrm{Bal}_{\mathrm{RC}}(n;\epsilon)$

Similar to the construction of $\mathrm{ENC}^1_{\mathrm{B_{RC}}(n;p)}$ and $\mathrm{DEC}^1_{\mathrm{B_{RC}}(n;p)}$ for $\mathrm{B_{RC}}(n;p)$ as presented in Subsection III-A, we obtain an efficient design $\mathrm{Bal}_{\mathrm{RC}}(n;\epsilon)$. For ease of exposition, we only highlight the main difference in redundancy in each encoding step as follows.

- In phase I, all rows in the array A are encoded to be ϵ-balanced via the encoding method in [23] and [25], that uses at most $c=\Theta(1)$ redundant bits for each row (refer to Theorem 2).
- In phase II, the encoder ensures that every column is ϵ-balanced via the divide and conquer algorithm as the case of p-bounded constraint. The total redundancy used to record all the swapping indices and the excluded columns (when n is not a power of 2) is $n\log n + O(n)$ bits. Let $\mathrm{Re}(n)$ be the sequence obtained by concatenating all binary representations of all swapping indices and the excluded columns.
- In phase III, we follow the method of Talyansky et al. [3] that lets $\mathrm{Re}(n)$ undergo the encoding process, recursively, to produce an array B in which all the rows and columns are ϵ-balanced. The array B, in turn, is appended to A so that the encoded array is of size $n\times n$. It was proved in [3] that the total redundancy for the encoder is at most $n\log n + O(n\log\log n) = n\log n + o(n\log n)$ bits.

It remains to point out the changes that need to be made in Phase II when n is not a power of 2, as what we have shown for the p-bounded constraint. Suppose that we have an array S of size $n_1\times(2k+1)$, which is already ϵ-balanced, i.e. $\mathrm{wt}(S) \in [(1/2-\epsilon)n_1(2k+1), (1/2+\epsilon)n_1(2k+1)]$. Here, we define L_S to consist of the first $(k+1)$ columns of S, whereas R_S consists of the remaining k columns of S. Observe that $n_1\epsilon$ is integers. Clearly, if L_S and R_S are both ϵ-balanced, then no swap is needed. Suppose, on the other hand, one of them is not ϵ-balanced. For simplicity, we assume that $\mathrm{wt}(S) \in [n_1(2k+1)/2, (1/2+\epsilon)n_1(2k+1)]$. The case that $\mathrm{wt}(S) \in [(1/2-\epsilon)n_1(2k+1), n_1(2k+1)/2]$ can be done similarly.

- (Case 1a) Suppose that L_S is not ϵ-balanced and $\mathrm{wt}(L_S) > (1/2+\epsilon)n_1(k+1)$. We then have $\mathrm{wt}(R_S) < (1/2+\epsilon)n_1k$. Similar to the p-bounded case, we set L'_S be the array obtained by removing the column with the minimum weight from L_S. Suppose that we remove the column j_1 for some $1 \leqslant j_1 \leqslant k+1$. We observe that L_S is not ϵ-balance, or $\mathrm{wt}(L'_S) > (1/2+\epsilon)n_1k$. We then swap the bits from L'_S and R_S. There must be an index t, $1 \leqslant t < n_1k$, such that $\mathrm{wt}(\mathrm{Swap}_t(R_S, L'_S)) = (1/2+\epsilon)n_1k$. For such an index t, we can show that the subarray, consisting of $\text{Swap}_t(L'_S, R_S)$ and the j_1th column, is also ϵ-balanced.

- (Case 1b) The case that L_S is not ϵ-balanced and $\text{wt}(L_S) < (1/2 - \epsilon)n_1(k+1)$ can be proved similarly.
- (Case 2a) Suppose that R_S is not ϵ-balanced and $\text{wt}(R_S) > (1/2 + \epsilon)n_1 k$. We then have $\text{wt}(L_S) < (1/2 + \epsilon)n_1(k+1)$. Similarly, we set L'_S be the array obtained by removing the column j_2 with the maximum weight from L_S. We observe that $\text{wt}(L'_S) < (1/2 + \epsilon)n_1 k$. We then swap the bits from L'_S and R_S, and there must be an index t, $1 \leqslant t < n_1 k$, such that $\text{wt}(\text{Swap}_t(R_S, L'_S)) = (1/2 + \epsilon)n_1 k$, and the subarray, consisting of $\text{Swap}_t(L'_S, R_S)$ and the j_2th column, is also ϵ-balanced.
- (Case 2b) The case that R_S is not ϵ-balanced and $\text{wt}(R_S) < (1/2 - \epsilon)n_1 k$ can be proved similarly.

We summarise the result as follows.

Theorem 3: Given $n, \epsilon > 0$, $\epsilon n \geqslant 1$, there exist an efficient encoder $\text{ENC}^1_{\text{Bal}_{\text{RC}}(n;\epsilon)}$ and a corresponding decoder $\text{DEC}^1_{\text{Bal}_{\text{RC}}(n;\epsilon)}$ for $\text{Bal}_{\text{RC}}(n;\epsilon)$ with at most $n \log n + o(n \log n)$ redundant bits.

In the next section, we show that for sufficiently large n, we can encode $\text{Bal}_{\text{RC}}(n;\epsilon)$ or encode $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$ with only one redundant bit.

IV. Efficient Encoders/Decoders for $\text{B}_{\text{RC}}(n;p)$ and $\text{Bal}_{\text{RC}}(n;\epsilon)$ via the Sequence Replacement Technique

The Sequence Replacement Technique (SRT) has been widely applied in the literature (for example, see [24], [25], [29], [30]). This is an efficient method for removing forbidden substrings from a source word. The advantage of this technique is that the complexity of encoder and decoder is very low, and they are also suitable for parallel implementation. In general, the encoder removes the forbidden strings and subsequently inserts its representation (which also includes the position of the substring) at predefined positions in the sequence. In our recent work [24], for codewords of length m, we enforced the almost-balanced weight-constraint over every subsequence consisting of ℓ consecutive bits where $\ell = \Omega(\log m)$.

Theorem 4 (Nguyen et al. [24]): Given p_1, p_2 where $0 \leqslant p_1 < 1/2 < p_2 \leqslant 1$, let $c = \min\{1/2 - p_1, p_2 - 1/2\}$. For $(1/c^2)\ln m \leqslant \ell \leqslant m$, there exists linear-time algorithm $\text{ENC} : \{0,1\}^{m-1} \to \{0,1\}^m$ such that for all $\boldsymbol{x} \in \{0,1\}^{m-1}$ if $\boldsymbol{y} = \text{ENC}(\boldsymbol{x})$ then $\text{wt}(\boldsymbol{y}) \in [p_1 m, p_2 m]$) and for every subsequence $\boldsymbol{w}$ consisting of ℓ consecutive bits of $\boldsymbol{y}$, $\text{wt}(\boldsymbol{w}) \in [p_1\ell, p_2\ell]$.

In this section, we show that, for sufficiently large n, the redundancy to encode (decode) binary data to (from) $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$ and $\text{Bal}_{\text{RC}}(n;\epsilon)$ for arbitrary $\epsilon \in (0, 1/2)$ can be reduced significantly to only a single bit via the SRT. Particularly, we provide two efficient encoders:

- The first encoder adapts the SRT (presented in [24]) with the antipodal matching (constructed in [11]) to encode arbitrary data to $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$ with at most $n + 3$ redundant bits.
- The second encoder, which is the main contribution of this work, modifies the SRT as presented in [24] to encode $\text{Bal}_{\text{RC}}(n;\epsilon)$ with only one redundant bit. Since $\text{Bal}_{\text{RC}}(n;\epsilon) \subset \text{B}_{\text{RC}}(n;p)$ for all $\epsilon \leqslant p - 1/2$, this can be apply to encode $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$ with one redundant bit as well.

A. SRT and Antipodal Matching for $\text{B}_{\text{RC}}(n;p)$ When $p > 1/2$

Recall that the encoders proposed in [11] can be used for constructing $\text{B}_{\text{RC}}(n;p)$ when $p > 1/2$, and the redundancy is roughly $2n$ (bits). In this section, we provide a linear-time encoder for $\text{B}_{\text{RC}}(n;p)$ where $p > 1/2$ with at most $(n+3)$ redundant bits. We have $p > 1/2$ in all our descriptions in this part.

Recall that for an array A of size $n \times n$, we use A_i to denote the ith row of A and A^j to denote the jth column of A. In addition, we use $A_{i;\langle j\rangle}$ to denote the sequence obtained by taking the first j entries of the row A_i and use $A^{i;\langle j\rangle}$ to denote the sequence obtained by taking the first j entries of the column A^i. For example, if

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}, \text{ then } A_{1;\langle 2\rangle} = ab, A^{3;\langle 2\rangle} = cf.$$

We now describe the detailed construction of the Encoder $\text{ENC}^2_{\text{B}_{\text{RC}}(n;p)}$, where $p > 1/2$.

Encoder $\text{ENC}^2_{\text{B}_{\text{RC}}(n;p)}$, where $p > 1/2$. Set $N = n^2 - (n+3)$, $\ell = n$, $p_1 = 0$, $p_2 = p$ and $c = p - 1/2$. According to Theorem 4, for sufficiently large n that $(1/c^2)\ln(n^2-n-2) \leqslant n \leqslant n^2 - n - 2$, there exists linear-time encoder, $\text{ENC}_{\text{seq}} : \{0,1\}^N \to \{0,1\}^{N+1}$ such that for all $\boldsymbol{x} \in \{0,1\}^N$ and $\boldsymbol{y} = \text{ENC}_{\text{seq}}(\boldsymbol{x})$ we have $\text{wt}(\boldsymbol{w}) \in [0, pn]$ for every subsequence $\boldsymbol{w}$ consisting of n consecutive bits of $\boldsymbol{y}$. In addition, we follow [11] to construct the antipodal matchings ϕ for sequences of length $n-1$.

INPUT: $\boldsymbol{x} \in \{0,1\}^N$
OUTPUT: $A \triangleq \text{ENC}^2_{\text{B}_{\text{RC}}(n;p)}(\boldsymbol{x}) \in \text{B}_{\text{RC}}(n;p)$ with $p > 1/2$

(I) Set $\boldsymbol{y} = \text{ENC}_{\text{seq}}(\boldsymbol{x}) \in \{0,1\}^{N+1}$. Suppose that $\boldsymbol{y} = y_1 y_2 \ldots y_{n^2-n-2}$.

(II) Fill $n^2 - n - 1$ bits of $\boldsymbol{y}$ to A row by row as follows.
- Set $A_i \triangleq y_{n(i-1)+1} \ldots y_{ni}$ for $1 \leqslant i \leqslant n-2$.
- Set $A_{n-1} \triangleq y_{n(n-2)+1} \ldots y_{n^2-n-2} *_1 *_2$, where $*_1, *_2$ are determined later.
- Suppose that $A_n = z_1 z_2 \ldots z_n$ where z_i is determined later.
- If $\text{wt}\Big(A_{n-1;\langle n-2\rangle}\Big) > p(n-2)$, flip all bits in $A_{n-1;\langle n-2\rangle}$ and set $*_1 = 1$, otherwise set $*_1 = 0$.

(III) For $1 \leqslant i \leqslant (n-1)$, we check the ith column:
- if $\text{wt}\Big(A^{i;\langle n-1\rangle}\Big) > pn$, set $z_i = 1$ and replace $A^{i;\langle n-1\rangle}$ with $\phi\Big(A^{i;\langle n-1\rangle}\Big)$
- Otherwise, set $z_i = 0$.

(IV) Check the nth row:
- If $\text{wt}\Big(A_{n;\langle n-1\rangle}\Big) > pn$, set $*_2 = 1$ and replace $A_{n;\langle n-1\rangle}$ with $\phi\Big(A_{n;\langle n-1\rangle}\Big)$
- Otherwise, set $*_2 = 0$.

(V) Check the nth column:
- If $\mathrm{wt}\left(A^{n;\langle n-1\rangle}\right) > pn$, set $z_n = 1$ and replace $A^{n;\langle n-1\rangle}$ with $\phi\left(A^{n,\langle n-1\rangle}\right)$.
- Otherwise, set $z_n = 0$.

(VI) Output A.

We have the following result.

Theorem 5: The Encoder $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}$ is correct. In other words, $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}(\boldsymbol{x}) \in \mathrm{B_{RC}}(n;p)$ with $p > 1/2$ for all $\boldsymbol{x} \in \{0,1\}^N$. The redundancy is $n+3$ (bits).

Proof: Let $A = \mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}(\boldsymbol{x})$. We first show that the weight of every column of A is at most pn. From Step (III) and Step (V), the encoder guarantees that the weights of n columns are at most pn. Although there is a replacement in the nth row in Step (IV), it does not affect the weight of any column. Indeed, from the definition of an antipodal matching, if $\mathrm{wt}(\boldsymbol{x}) > n/2$, then $\phi(\boldsymbol{x})$ has all its 1's in positions where $\boldsymbol{x}$ has 1's and $\mathrm{wt}(\phi(\boldsymbol{x})) \leqslant n/2$. Therefore, whenever the encoder performs replacement step in any row (or respectively any column), it does not increase the weight of any column (or respectively any row). Therefore, we conclude that $\mathrm{wt}\left(A^i\right) \leqslant pn$ for all $1 \leqslant i \leqslant n$.

We now show that the $\mathrm{wt}\left(A_i\right) \leqslant pn$ for all $1 \leqslant i \leqslant n$. From Step (IV), we observe that the nth row satisfies the weight-constraint. As mentioned above, during Step (III) and Step (V), whenever the encoder performs replacement step in any column, it does not increase the weight of any row, i.e. the weight-constraint is preserved over the first $(n-2)$ rows that is guaranteed by the Encoder $\mathrm{ENC_{seq}}$ from Step (I). It remains to show that the $(n-1)$th row satisfies the weight constraint with the determined values of $*_1, *_2$. Indeed, from Step (II), if $*_1 = 0$, we have:

$$\mathrm{wt}\left(A_{n-1}\right) \leqslant p(n-2)+1 = pn+(1-2p) \leqslant pn.$$

Otherwise,

$$\mathrm{wt}\left(A_{n-1}\right) < (1-p)(n-2)+2 < pn \text{ for all } n > 2/(2p-1).$$

In conclusion, we have $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}(\boldsymbol{x}) \in \mathrm{B_{RC}}(n;p)$ for all $\boldsymbol{x} \in \{0,1\}^m$. Since $N = n^2-(n+3)$, the redundancy of our encoder is then $n^2-N = n+3$ (bits). ∎

Remark 1: We now discuss the lower bound for n so that the encoding algorithm works. It requires $(1/c^2)\ln(n^2-n-2) \leqslant n \leqslant n^2-n-2$ where $c = p-1/2$. Since $n^2-n-2 \geqslant n$ for all $n \geqslant 3$ and $\ln(n^2-n-2) \leqslant \ln n^2 = 2\ln n$, a simple lower bound can be described as $n \geqslant 3$ and $n/\ln n \geq 1/(p-1/2)^2$.

For completeness, we describe the corresponding decoder $\mathrm{DEC}^2_{\mathrm{B_{RC}}(n;p)}$ as follows. Recall that the positions of redundant bits are fixed by our encoder that includes all n bits of the last row A_n and the last two bits of A_{n-1} (refer to Step II in the construction of the encoder $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}$).

Decoder, $\mathrm{DEC}^2_{\mathrm{B_{RC}}(n;p)}$.

INPUT: $A \in \mathrm{B_{RC}}(n;p)$
OUTPUT: $\boldsymbol{x} \triangleq \mathrm{DEC}^2_{\mathrm{B_{RC}}(n;p)}(A) \in \{0,1\}^N$, where $N = n^2-n-3$

(I) Decode the nth column, A^n. If the last bit is 1, flip it to 0 and replace $A^{n;\langle n-1\rangle}$ with $\phi\left(A^{n;\langle n-1\rangle}\right)$. Otherwise, proceed to the next step.

(II) Decode the nth row, A_n. Check the last bit in A_{n-1}, if it is 1, flip it to 0 and replace $A_{n;\langle n-1\rangle}$ with $\phi\left(A_{n;\langle n-1\rangle}\right)$. Otherwise, proceed to the next step.

(III) Decode the $(n-1)$th row, A_{n-1}. Check the second last bit in A_{n-1}, if it is 1, flip all the bits in A_{n-1}. Otherwise, proceed to the next step.
Suppose the nth row is now $A_n = z_1z_2\ldots z_n$.

(IV) For $1 \leqslant i \leqslant (n-1)$, we decode the ith column, i.e. A^i, as follows. If $z_i = 1$, replace $A^{i;\langle n-1\rangle}$ with $\phi\left(A^{i,\langle n-1\rangle}\right)$.

(V) Set $\boldsymbol{y} \triangleq A_1A_2\ldots A_{n-2}A_{n-1;\langle n-2\rangle} \in \{0,1\}^{n^2-n-2}$.

(VI) Output $\boldsymbol{x} \triangleq \mathrm{DEC_{seq}}(\boldsymbol{y}) \in \{0,1\}^{n^2-n-3}$.

Complexity analysis. For $n \times n$ arrays, it is easy to verify that encoder $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}$ and decoder $\mathrm{DEC}^2_{\mathrm{B_{RC}}(n;p)}$ have linear-time complexity. Particularly, there are at most $n+2$ replacements, each replacement is done over sequence of length n, and the complexity of encoder/decoder $\mathrm{ENC_{seq}}, \mathrm{DEC_{seq}}$ is linear over codeword length $N = n^2-n-3 = \Theta(n^2)$. We conclude that the running time of encoder $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}$ and decoder $\mathrm{DEC}^2_{\mathrm{B_{RC}}(n;p)}$ is $\Theta(n^2)$ which is linear in the message length $N = n^2-n-3$.

Remark 2: In [3], Talyansky et al. studied the *t-conservative arrays constraint*, where every row has at least t transitions $0 \to 1$ or $1 \to 0$ for some $t \leqslant n/(2\log n) - O(\log n) < n/2$. Such a constraint is equivalent to the p-bounded constraint in a weaker condition, where the weight constraint is enforced in every row only. Indeed, for a sequence $\boldsymbol{x} = x_1x_2\ldots x_n \in \{0,1\}^n$, consider the *differential of* $\boldsymbol{x}$, denoted by $\mathrm{Diff}(\boldsymbol{x})$, which is a sequence $\boldsymbol{y} = y_1y_2\ldots y_n \in \{0,1\}^n$, where $y_1 = x_1$ and $y_i = x_i - x_{i-1} \pmod 2$ for $2 \leqslant i \leqslant n$. We then observe that $\boldsymbol{x}$ has at least t transitions if and only if the weight of $\mathrm{Diff}(\boldsymbol{x})$ is at least t. In addition, the constraint problem where the weight in every row is at least t where $t < n/2$ is equivalent to the constraint problem where the weight in every row is at most t where $t > n/2$. Therefore, one may modify the construction of our proposed encoder $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}$ (i.e for $p = 1-1/(2\log n)) > 1/2$) to construct binary arrays such that there are at least t transitions in every row and every column with at most $n+3$ redundant bits. When the weight-constraint is only required on rows, only Step (I) in $\mathrm{ENC}^2_{\mathrm{B_{RC}}(n;p)}$ is sufficient and $N = n^2-1$. Although there is no improvement in the redundancy (the encoder in [3] also use only one redundant bit), our encoder can be applied for a larger range of t, where $t \leqslant n/c$ for any $c > 2$.

B. Modified SRT to Encode $\mathrm{Bal_{RC}}(n;\epsilon)$ With One Redundant Bit

In this section, we show that, for sufficient n, the redundancy to encode (decode) binary data to (from) $\mathrm{Bal_{RC}}(n;\epsilon)$ can be further reduced from $\Theta(n)$ bits (from the encoding method in Section III-B) to only a single bit via the SRT.

Similar to the coding method in [24], the original data of length (n^2-1) is prepended with 0. We then also remove all the *forbidden strings*, and append the replacement strings (starting with *marker* 1). To ensure that the encoding process terminates, replacement strings are of shorter lengths compared to the forbidden strings.

On the other hand, the differences in this work are as follows. There are two types of forbidden strings: those comprising consecutive bits (to ensure the weight constraint over the rows) and those comprising bits that are of distance n bits apart (to ensure the weight constraint over the columns). Consequently, we have two types of markers.

Definition 7: For a binary sequence $\boldsymbol{x} = x_1x_2\ldots x_m$, a subsequence $\boldsymbol{y}$ of $\boldsymbol{x}$ is said to be (ℓ,ϵ)-*r-forbidden* if $\boldsymbol{y} = x_ix_{i+1}\ldots x_{i+\ell-1}$ for some i and $\boldsymbol{y}$ is not ϵ-balanced. On the other hand, a subsequence $\boldsymbol{z}$ of $\boldsymbol{x}$ is said to be (ℓ,ϵ)-*c-forbidden* if $\boldsymbol{z} = x_jx_{j+n}\ldots x_{j+(\ell-1)n}$ for some j and $\boldsymbol{z}$ is not ϵ-balanced.

Given $\epsilon > 0$, let $\mathbf{F}(\ell,\epsilon)$ denote the set of all forbidden sequences of size ℓ, that are not ϵ-balanced. The following theorem provides an upper bound on the size of $\mathbf{F}(\ell,\epsilon)$.

Theorem 6: [24, Theorem 5] For $\epsilon > 0, m \geqslant 16$ and $\ell \leqslant m$ such that $(1/\epsilon^2)\ln m \leqslant \ell$, let $k = \ell - 3 - \log m$, there exists an one-to-one map $\Psi : \mathbf{F}(\ell,\epsilon) \rightarrow \{0,1\}^k$.

The look-up table for $\Psi : \mathbf{F}(\ell,\epsilon) \rightarrow \{0,1\}^k$, which is roughly of size 2^ℓ, is needed for our encoding and decoding algorithms. To obtain efficient algorithms whose running time is linear in m, we set $\ell = \alpha \ln m = \Theta(\log m)$, for some fixed $\alpha \geqslant 1/\epsilon^2$.

Note that, when all forbidden strings have been removed, the length of the current encoded sequence is strictly smaller than n^2. Similar to the coding methods in [24] and [30], we introduce the *extension phase* procedure to append bits to obtain an encoded sequence of length n^2 while the weight constraint is still preserved. Crucial to the correctness of our encoding algorithm is the following lemma.

Lemma 2: Given $\epsilon > 0$, integers n, ℓ, s, where $n = \ell s$, $n\epsilon \geqslant 2$. Set $\epsilon' = \epsilon/2$ and suppose that $\boldsymbol{x} \in \{0,1\}^{n-1}$ where every window of size ℓ of $\boldsymbol{x}$ is ϵ'-balanced. We then have $\boldsymbol{x}0$ and $\boldsymbol{x}1$ that are both ϵ-balanced.

Proof: Since every window of size ℓ of $\boldsymbol{x}$ is ϵ'-balanced, we then have

$$\begin{aligned}\mathrm{wt}(\boldsymbol{x}1) \geqslant \mathrm{wt}(\boldsymbol{x}0) &\geqslant (1/2-\epsilon')\ell s - 1\\ &= (1/2-\epsilon)n + n\epsilon/2 - 1\\ &\geqslant (1/2-\epsilon)n.\end{aligned}$$

Similarly, we have

$$\begin{aligned}\mathrm{wt}(\boldsymbol{x}0) \leqslant \mathrm{wt}(\boldsymbol{x}1) &\leqslant (1/2+\epsilon')\ell s + 1\\ &= (1/2+\epsilon)\ell s + (1 - \epsilon\ell s/2)\\ &= (1/2+\epsilon)n + (1-\epsilon n/2)\\ &\leqslant (1/2+\epsilon)n.\end{aligned}$$

Thus, $\boldsymbol{x}0$ or $\boldsymbol{x}1$ are both ϵ-balanced. ■

We now present a linear-time algorithm $\textsc{Enc}^2_{\mathrm{Bal}_{\mathrm{RC}}(n;\epsilon)}$ to encode $\mathrm{Bal}_{\mathrm{RC}}(n;\epsilon)$ that incurs at most one redundant bit.

Encoder $\textsc{Enc}^2_{\mathrm{Bal}_{\mathrm{RC}}(n;\epsilon)}$. Given $n, \epsilon > 0$, we set $m = n^2, \epsilon' = \epsilon/2, \ell = \lceil \alpha \log_e m \rceil$ for some fixed number $\alpha \geqslant 1/\epsilon'^2$. The source sequence $\boldsymbol{x} \in \{0,1\}^{n^2-1}$. For simplicity, we assume $\log n$ is an integer, and $n = \ell s$ for some integer s. According to Theorem 6, there exists an one-to-one map $\Psi : \mathbf{F}(\ell,\epsilon') \rightarrow \{0,1\}^k$, where $k = \ell - 3 - \log m = \ell - 3 - 2\log n$. Here, $\mathbf{F}(\ell,\epsilon')$ denotes the set of all forbidden sequences of size ℓ, that are not ϵ'-balanced and can be mapped one-to-one to binary sequences of length $\ell - 3 - 2\log n$.

The algorithm contains three phases: *initial phase*, *replacement phase* and *extension phase*. Particularly, the extension phase includes *row extension* and *array extension*.

Initial phase. The source sequence $\boldsymbol{x} \in \{0,1\}^{n^2-1}$ is prepended with 0, to obtain $\boldsymbol{c} = 0\boldsymbol{x} \in \{0,1\}^{n^2}$. The encoder scans $\boldsymbol{c}$ and if there is no forbidden subsequence, neither (ℓ,ϵ')-r-forbidden or (ℓ,ϵ')-c-forbidden, it outputs $\Phi^{-1}(\boldsymbol{c})$, which is an array of size $n \times n$. Otherwise, it proceeds to the replacement phase. Observe that if there is no (ℓ,ϵ')-r-forbidden or (ℓ,ϵ')-c-forbidden in all rows and all columns then all rows and all columns are ϵ'-balanced, and since $\epsilon' = \epsilon/2 < \epsilon$, all rows and all columns are then ϵ-balanced.

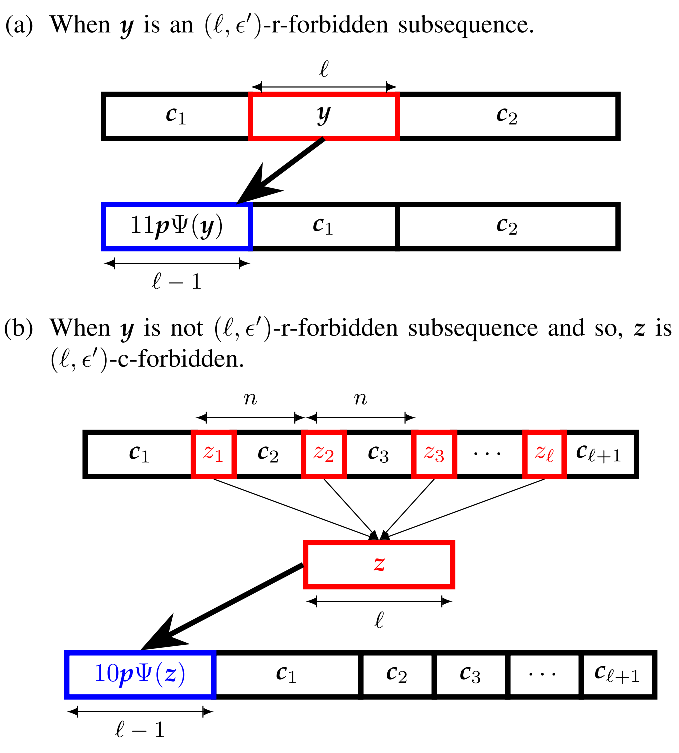
Replacement phase. Let the current word $\boldsymbol{c}$ of length N_0. In the beginning, $N_0 = n^2$. The encoder searches for the forbidden subsequences. Suppose the first forbidden subsequence starts at x_i for some $1 \leqslant i \leqslant N_0 - \ell + 1$. Let $\boldsymbol{p}$ be the binary representation of the index i of length $2\log n$. Let $\boldsymbol{y} = x_ix_{i+1}\ldots x_{i+\ell-1}$ and $\boldsymbol{z} = x_ix_{i+n}\ldots x_{i+(\ell-1)n}$.

- If $\boldsymbol{y}$ is (ℓ,ϵ')-r-forbidden, the encoder sets $\mathrm{R} = 11\boldsymbol{p}\Psi(\boldsymbol{y})$. It then removes $\boldsymbol{y}$ from $\boldsymbol{c}$ and prepends R to $\boldsymbol{c}$. The encoder repeats the replacement phase.
- If $\boldsymbol{y}$ is not (ℓ,ϵ')-r-forbidden, and $\boldsymbol{z}$ is (ℓ,ϵ')-c-forbidden, the encoder sets $\mathrm{R} = 10\boldsymbol{p}\Psi(\boldsymbol{z})$. It then removes $\boldsymbol{z}$ from $\boldsymbol{c}$ and prepends R to $\boldsymbol{c}$. The encoder repeats the replacement phase.

The encoder exits the replacement phase and proceeds to the extension phase if, after some replacement, the current sequence $\boldsymbol{c}$ contains no (ℓ,ϵ')-r-forbidden (or (ℓ,ϵ')-c-forbidden) subsequence, or the current sequence is of length $n/2$. Otherwise, the encoder repeats the replacement phase. Note that such a replacement operation reduces the length of the sequence by one, since we remove a subsequence of ℓ bits and replace it by $2 + 2\log n + k = 2 + 2\log n + (\ell - 3 - 2\log n) = \ell - 1$ (bits). Therefore, this procedure is guaranteed to terminate. We illustrate the idea of the replacement phase through Figure 2.

Extension phase. If the length of the current sequence $\boldsymbol{c}$ is N_0 where $N_0 < n$, the encoder appends a suffix of length $N_1 = n^2 - N_0$ to obtain a sequence of length n^2. Note that at the end of the replacement phase, the length of the current sequence is at least $n/2$. Suppose that $N_0 = n \times q + r$ where $0 \leqslant q < n$, $0 \leqslant r < n$. If we fill the bits in $\boldsymbol{c}$ to an array of size $n \times n$, we can fill q rows and the $(q+1)$th row includes r bits. From the replacement phase, in the worst case scenario, we have $N_0 = n/2, q = 0, r = n/2$. The extension phase includes two steps: the row extension step $\big($to fulfill the $(q+1)$th row$\big)$ and the array extension step $\big($to fulfill the remaining $(n-q-1)$ rows$\big)$.

Row extension. The encoder fulfills the $(q+1)$th row as follows.



(a) When $\boldsymbol{y}$ is an (ℓ, ϵ')-r-forbidden subsequence.
ℓ
$\boldsymbol{c}_1$
$\boldsymbol{y}$
$\boldsymbol{c}_2$
$11\boldsymbol{p}\Psi(\boldsymbol{y})$
$\boldsymbol{c}_1$
$\boldsymbol{c}_2$
$\ell-1$
(b) When $\boldsymbol{y}$ is not (ℓ, ϵ')-r-forbidden subsequence and so, $\boldsymbol{z}$ is (ℓ, ϵ')-c-forbidden.
n
n
$\boldsymbol{c}_1$ z_1 $\boldsymbol{c}_2$ z_2 $\boldsymbol{c}_3$ z_3 $\cdots$ z_ℓ $\boldsymbol{c}_{\ell+1}$
$\boldsymbol{z}$
ℓ
$10\boldsymbol{p}\Psi(\boldsymbol{z})$
$\boldsymbol{c}_1$ $\boldsymbol{c}_2$ $\boldsymbol{c}_3$ $\cdots$ $\boldsymbol{c}_{\ell+1}$
$\ell-1$

Fig. 2. (ℓ, ϵ')-r-forbidden and (ℓ, ϵ')-c-forbidden replacement.

- If $N_0 = n/2$, i.e. $q = 0$, $r = n/2$, the encoder simply concatenate the sequence $\boldsymbol{c}$ and its complement $\bar{\boldsymbol{c}}$ to obtain the first row. In this case, it is easy to see that the row is balanced.
- If $N_0 > n/2 > \ell$, we then observe that the output sequence from the replacement phase, $\boldsymbol{c}$, contains no (ℓ, ϵ')-r-forbidden subsequence of length ℓ. Let $\boldsymbol{w}$ be the suffix of length ℓ of $\boldsymbol{c}$. The encoder simply appends $\boldsymbol{w}$ repeatedly until the $(q+1)$th row is fulfilled. Formally, let j be the smallest integer such that $\boldsymbol{c}' = \boldsymbol{c}\boldsymbol{w}^j$ is of length greater than $(q+1)n$. The encoder outputs the prefix of length $(q+1)n$ of $\boldsymbol{c}'$. In this case, it is also easy to verify that, after the row extension phase, the $(q+1)$th row is ϵ'-balanced. Since $\boldsymbol{c}$ does not contain any (ℓ, ϵ')-r-forbidden subsequence, it remains to show that there is no (ℓ, ϵ')-r-forbidden subsequence in the suffix $\boldsymbol{w}^j$. It is easy to see that repeating the vector $\boldsymbol{w}$ clearly satisfies the constraint since every subsequence of size ℓ generated in this manner is a cyclic shift of the vector $\boldsymbol{w}$, and since $\boldsymbol{w}$ is ϵ'-balanced, there is no (ℓ, ϵ')-r-forbidden subsequence.

Array extension. After the row extension step, the array is of size $(q+1) \times n$, where $0 \leqslant q < n$, and every row is ϵ'-balanced.

- If $q+1 = n$ or $q = n-1$, the encoder simply outputs this array. Since every row is ϵ'-balanced, which is also ϵ-balanced, it remains to show that every column is ϵ-balanced. From the replacement step, every window of size ℓ of the first $(n-1)$ bits in every column is ϵ'-balanced. According to Lemma 2, we then have every column is ϵ-balanced.
- If $q+1 \leqslant n/2$, the encoder simply fills in the next $(q+1)$ rows with the complement of the current array. After that the encoders fills the remaining rows alternately with balanced sequences $\boldsymbol{w} = 0101\ldots = (01)^{n/2}$, and $\boldsymbol{w}' = 1010\ldots = (10)^{n/2}$. In this case, it is easy to show that every row and every column is ϵ-balanced.
- If $q+1 > n/2$, similar to the process in the row extension step, for each column i, the encoder sets $\boldsymbol{y}_i$ to be the sequence obtained by the first q bits (i.e. except the bit in the $(q+1)$th row) and $\boldsymbol{w}_i$ to be the suffix of length ℓ of $\boldsymbol{y}_i$. It then appends $\boldsymbol{w}_i$ repeatedly until the ith column is fulfilled. Similar to the proof in the row extensive phase, for every column, the sequence obtained by $(n-1)$ bits, except the bit in the $(q+1)$th row, is ϵ'-balanced. Again, according to Lemma 2, every column is ϵ-balanced. Note that all $(n-q-1)$ rows, that have been fulfilled, are actually repetitions of some previous rows (i.e. the jth column is filled exactly as the $j-\ell-1$th column), therefore, they are all ϵ'-balanced, and hence, are also ϵ-balanced.

The following result is then immediate.

Theorem 7: The Encoder $\text{ENC}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}$ is correct, i.e. $\text{ENC}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}(\boldsymbol{x}) \in \text{Bal}_{\text{RC}}(n;\epsilon)$ for all $\boldsymbol{x} \in \{0,1\}^{n^2-1}$. The redundancy of the proposed encoder is one bit.

For completeness, we describe the corresponding decoder $\text{DEC}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}$ as follows.

Decoder $\text{DEC}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}$.

INPUT: $A \in \text{Bal}_{\text{RC}}(n;\epsilon)$
OUTPUT: $\boldsymbol{x} \triangleq \text{DEC}^2_{\text{Bal}_{\text{RC}}(n;\epsilon)}(A) \in \{0,1\}^{n^2-1}$

Decoding procedure. From an array A of size $n \times n$, the decoder first obtains the binary sequence $\boldsymbol{x} = \Phi(A)$ of length n^2. The decoder scans from left to right. If the first bit is 0, the decoder simply removes 0 and identifies the last (n^2-1) bits are source data. On the other hand, if it starts with 11, the decoder takes the prefix of length $(\ell-1)$ and concludes that this prefix is obtained by a replacement of (ℓ, ϵ')-r-forbidden subsequence. In other words, the prefix is of the form $11\boldsymbol{p}\Psi(\boldsymbol{y})$, where $\boldsymbol{p}$ is of length $2\log n$ and $\Psi(\boldsymbol{y})$ is of length k. The decoder removes this prefix, adds the subsequence $\boldsymbol{y} = \Psi^{-1}(\Psi(\boldsymbol{y}))$ into position i, which takes $\boldsymbol{p}$ as the binary representation. On the other hand, if it starts with 10, the decoder takes the prefix of length $(\ell-1)$ and concludes that this prefix is obtained by a replacement of (ℓ, ϵ')-c-forbidden subsequence. In other words, the prefix is of the form $10\boldsymbol{p}\Psi(\boldsymbol{z})$, where $\boldsymbol{p}$ is of length $2\log n$ and $\Psi(\boldsymbol{z})$ is of length k. The decoder removes this prefix, and adds the forbidden subsequence $\boldsymbol{z} = \Psi^{-1}(\Psi(\boldsymbol{z}))$ into position i, which takes $\boldsymbol{p}$ as the binary representation. It terminates when the first bit is 0, and simply takes the following (n^2-1) bits as the source data.

We illustrate the idea of the extension phase through the following example.

Example 3: Consider $n = 6, \ell = 3, \epsilon = 1/6$, i.e. the weight of every row and every column is within $[2, 4]$. The first example considers the worst case scenario.

$$\begin{pmatrix} 1 & 1 & 0 & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

In this example, $N_0 = n/2 = 3$ and $q = 0$. The bits in blue denote the complement of the first $n/2$ bits in the row extension step, the bits in green denote the complement of the first row in the array extension step since $q+1 = 1 < n/2$, and the last three rows are filled by sequences $\boldsymbol{w} = 010101$ and $\boldsymbol{w'} = 101010$ alternately.

Another example is as follows.

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

In this example, $N_0 = 25$ and $q = 4$. The encoder first fulfills the $(q+1)$th or the fifth row, and then proceeds to the array extension. Here, the bits in blue denote the suffix of length ℓ bits before the row extension step, and the bits in red denote the process that the encoder repeats the suffix until the fifth row is fulfilled, and finally the last row is filled as the repetition of the second row. Throughout the two examples, we can verify that every row and every column is ϵ-balanced.

Remark 3: We now discuss the lower bound for n so that the encoding algorithm works. Given $\epsilon \in (0, 1/2)$, we require $n^2 \geqslant 1/\epsilon'^2 \ln n^2 = 8/\epsilon^2 \ln n$ and $n\epsilon \geqslant 2$. Since $n \gg \ln n$, a simple lower bound would be $n \geqslant 8/\epsilon^2$. Furthermore, when $p > 1/2$, if we set $\epsilon = p - 1/2$ then $\text{Bal}_{\text{RC}}(n;\epsilon) \subset \text{B}_{\text{RC}}(n;p)$. Hence, one may use our encoding method for $\text{Bal}_{\text{RC}}(n;\epsilon)$ to encode $\text{B}_{\text{RC}}(n;p)$, which uses one redundant bit. This improves the redundancy of the encoder in Subsection IV-A, which incurs $n+3$ redundant bits.

V. 2D Sliding-Window Constrained Codes

In this section, we are interested in the problem of designing efficient coding methods that encode (decode) binary data to (from) $\text{B}_{\text{SW}}(n,m;p)$ and $\text{Bal}_{\text{SW}}(n,m;\epsilon)$. Similar to the case of constructing 2D RC constrained codes $\text{B}_{\text{RC}}(n;p)$ and $\text{Bal}_{\text{RC}}(n;\epsilon)$, the challenge in coding design is to find an efficient method to enforce the weight constraint in every window so that changing the weight of a window does not violate the weight-constraint in previously coded windows.

Our main results in this section are summarised as follows.

- We use the construction of antipodal matching (as discussed in Section II-B and Section IV-A) to encode (decode) binary data to (from) $\text{B}_{\text{SW}}(n,m;p)$ with at most n redundant bits when $m = n - \Theta(1)$ and $p \geqslant 1/2$. The construction can be extended to obtain capacity-approaching encoder with at most $o(n^2)$ redundant bits when $m = n - o(n)$.
- We use SRT to encode (decode) binary data to (from) $\text{Bal}_{\text{SW}}(n,m;\epsilon)$. For sufficiently large n, this method incurs at most one redundant bit.

We first recall the sets $\text{B}_{\text{SW}}(n,m;p)$ and $\text{Bal}_{\text{SW}}(n,m;\epsilon)$. Given n,m,p,ϵ, where $m \leqslant n, \epsilon \in [0,1/2], p \in [0,1]$ we set

$$\text{B}_{\text{SW}}(n,m;p) \triangleq \Big\{ A \in \boldsymbol{A}_n : \text{every window } W \text{ of } A \text{ of size } m \times m \text{ are } p\text{-bounded} \Big\},$$

$$\text{Bal}_{\text{SW}}(n,m;\epsilon) \triangleq \Big\{ A \in \boldsymbol{A}_n : \text{every window } W \text{ of } A \text{ of size } m \times m \text{ are } \epsilon\text{-balanced} \Big\}.$$

Example 4: Consider $n = 4, m = 2, p = 1/4$ and two given arrays A and B as follows. We observe that $A \in \text{B}_{\text{RC}}(n;p)$, however $A \notin \text{B}_{\text{SW}}(n,m;p)$ since there is a window of size 2×2 (highlighted in red), which is not p-bounded. On the other hand, we have $B \in \text{B}_{\text{SW}}(n,m;p)$, and $B \notin \text{B}_{\text{RC}}(n;p)$ as the first column of B (highlighted in blue) is not p-bounded.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A. Antipodal Matching for $\text{B}_{\text{SW}}(n,m;p)$

Suppose that $m = n - k$ for some constant number $k = \Theta(1)$. Observe that $\text{B}_{\text{SW}}(n,m;1/2) \subset \text{B}_{\text{SW}}(n,m;p)$ for all $p \geqslant 1/2$. Hence, we aim to encode (decode) binary data to (from) $\text{B}_{\text{SW}}(n,m;1/2)$ with at most n redundant bits, i.e. for any array $A \in \text{B}_{\text{SW}}(n,m;1/2)$, the weight of every window of size $m \times m$ is at most $m^2/2$. For simplicity, we suppose that m is even and $2(k+1)^2 \leqslant n$.

We follow [11] to construct the antipodal matchings ϕ_1 for sequences of length m^2 and ϕ_2 for sequences of length $m^2 - m$. In other words, $\phi_1 : \{0,1\}^{m^2} \rightarrow \{0,1\}^{m^2}$ and $\phi_2 : \{0,1\}^{m^2-m} \rightarrow \{0,1\}^{m^2-m}$ such that for arbitrary $\boldsymbol{x} \in \{0,1\}^{m^2}$ and $\boldsymbol{y} \in \{0,1\}^{m^2-m}$:

- $\text{wt}(\phi_1(\boldsymbol{x})) = n - \text{wt}(\boldsymbol{x})$, and $\text{wt}(\phi_2(\boldsymbol{y})) = n - \text{wt}(\boldsymbol{y})$,
- If $\text{wt}(\boldsymbol{x}) > m^2/2$ then $\phi_1(\boldsymbol{x})$ has all its 1's in positions where $\boldsymbol{x}$ has 1's. In other words, suppose $\boldsymbol{x} = x_1 x_2 \ldots x_{m^2}$ and $\boldsymbol{z} = \phi_1(\boldsymbol{x}) = z_1 z_2 \ldots z_{m^2}$, then $z_i = 1$ implies $x_i = 1$ for $1 \leqslant i \leqslant m^2$. We have the same argument for $\boldsymbol{y}$.
- $\phi_1(\phi_1(\boldsymbol{x})) = \boldsymbol{x}$ and $\phi_2(\phi_2(\boldsymbol{y})) = \boldsymbol{y}$.

We now describe the detailed construction of the sliding-window p-bounded encoder, $\text{Enc}_{\text{B}_{\text{SW}}(n,m;p)}$ when $p \geqslant 1/2$.

SW p-bounded encoder, $\text{Enc}_{\text{B}_{\text{SW}}(n,m;p)}$.

Input: $\boldsymbol{x} \in \{0,1\}^{n^2-n}$
Output: $A \triangleq \text{Enc}_{\text{B}_{\text{SW}}(n,m;p)}(\boldsymbol{x}) \in \text{B}_{\text{SW}}(n,m;1/2)$, where $\text{B}_{\text{SW}}(n,m;1/2) \subset \text{B}_{\text{SW}}(n,m;p)$

(I) Fill $n^2 - n$ bits of $\boldsymbol{x}$ to A row by row to obtain a subarray of size $(n-1) \times n$ and suppose the last row of A is $A_n = y_1 y_2 \ldots y_n$.
(II) Observe that there are $(k+1)^2$ windows of size $m \times m$ that we need to enforce the weight constraint. We set

the order of the windows as row by row and let W_i be the ith window of A for $1 \leqslant i \leqslant (k+1)^2$.

(III) Using ϕ_1 for W_i where $1 \leqslant i \leqslant k(k+1)$:
- If $\mathrm{wt}(W_i) > m^2/2$, set $y_{2i-1} = 1$ and replace the entire subarray W_i with $\phi_1(W_i)$.

(IV) Using ϕ_2 for $(k+1)$ windows of size $(m-1)\times m$:
- For $k(k+1)+1 \leqslant i \leqslant (k+1)^2$, set C_i be the subarray obtained by removing the last row of W_i. In other words, C_i is of size $(m-1)\times m$ which does not include the bits of the last row A_n.
- If $\mathrm{wt}(C_i) > (m^2-m)/2$, set $y_{2i-1} = 1$ and replace the entire subarray C_i with $\phi_2(C_i)$.

(V) Filling the remaining bit of the nth row:
- Set $y_{2i} = \overline{y_{2i-1}}$ for $1 \leqslant i \leqslant (k+1)^2$. We then obtain $y_1y_2\ldots y_{2(k+1)^2}$ as a balanced sequence.
- Fill the remaining bits of the nth row with 0.

(VI) Output A.

Theorem 8: The Encoder $\textsc{Enc}_{\mathrm{B_{SW}}(n,m;p)}$ is correct. In other words, $\textsc{Enc}_{\mathrm{B_{SW}}(n,m;p)}(\boldsymbol{x}) \in \mathrm{B_{SW}}(n,m;1/2)$ for all $\boldsymbol{x} \in \{0,1\}^{n^2-n}$. The redundancy is n (bits).

Proof: Suppose that $A = \textsc{Enc}_{\mathrm{B_{SW}}(n,m;p)}(\boldsymbol{x})$ for some $\boldsymbol{x} \in \{0,1\}^{n^2-n}$. We now show that every window of size $m\times m$ has weight at most $m^2/2$. We set the order of the windows as row by row and let W_i be the ith window of A for $1 \leqslant i \leqslant (k+1)^2$.

We first show that the weight of W_i is at most $m^2/2$ for $1 \leqslant i \leqslant k(k+1)$. From step (III), suppose that a window W_i satisfies the weight constraint and the encoder proceeds to replace some window W_j with $\phi_1(W_j)$ where W_j has some overlapping bits with W_i and $\mathrm{wt}(W_j) > m^2/2$. Since $\phi_1(W_j)$ has all its 1's in positions where W_j has 1's and $\mathrm{wt}(\phi_1(W_j)) \leqslant m^2/2$. Therefore, whenever the encoder performs replacement in W_j, it does not increase the weight of W_i, the '0' bits in W_i will not change to '1', only the '1' bits in W_i will either stays as 1 or change to 0. Hence, it does not violate the weight constraint in W_i. Thus, at the end of step (III), the encoder ensures that the weight of W_i is at most $m^2/2$ for $1 \leqslant i \leqslant k(k+1)$. Similarly, at the end of step (IV), we have the weight of C_i is at most $(m^2-m)/2$ for $k(k+1)+1 \leqslant i \leqslant (k+1)^2$.

It remains to show that the weight of W_i is at most $m^2/2$ for $k(k+1)+1 \leqslant i \leqslant (k+1)^2$. Observe that, from step (V), every m consecutive bits of the last row A_n form a balanced sequence (here m is even). For $k(k+1)+1 \leqslant i \leqslant (k+1)^2$, each W_i is the concatenation of C_i (the weight of C_i is at most $(m^2-m)/2$) and m consecutive bits of the last row A_n (which is a balanced sequence). Thus, the weight of W_i is also at most $(m^2-m)/2 + m/2 = m^2/2$. ■

For completeness, we describe the corresponding decoder $\textsc{Dec}_{\mathrm{B_{SW}}(n,m;p)}$ as follows.

SW p-bounded decoder, $\textsc{Dec}_{\mathrm{B_{SW}}(n,m;p)}$.

Input: $A \in \mathrm{B_{SW}}(n,m;p)$
Output: $\boldsymbol{x} \triangleq \textsc{Dec}_{\mathrm{B_{SW}}(n,m;p)}(A) \in \{0,1\}^{n^2-n}$

(I) Suppose that $A_n = y_1y_2\ldots y_n$. We set the order of the windows as row by row and let W_i be the ith window of A of size $m\times m$ for $1 \leqslant i \leqslant (k+1)^2$. For $k(k+1)+1 \leqslant i \leqslant (k+1)^2$, set C_i to be the subarray obtained by removing the last row of W_i. In other words, C_i is of size $(m-1)\times m$.

(II) For $1 \leqslant i \leqslant k(k+1)$, if $y_{2i-1} = 1$, replace W_i with $\phi_1(W_i)$.

(III) For $k(k+1)+1 \leqslant i \leqslant (k+1)^2$, if $y_{2i-1} = 1$, replace C_i with $\phi_2(C_i)$.

(IV) Set A' be the array obtained by the first $(n-1)$ rows of the current array.

(V) Output $\boldsymbol{x} \triangleq \Phi^{-1}(A') \in \{0,1\}^{n^2-n}$.

We illustrate the idea of the encoding process through the following example.

Example 5: Consider $n = 9, m = 8$, i.e. the weight of every window of size 8×8 is at most 32. We construct two antipodal matchings: ϕ_1 for sequences of length 64 and ϕ_2 for sequences of length 56. There are four windows of A to enforce the weight constraint and the last row $A_9 = y_1y_2\ldots y_9$ is for the redundant bits.

$$A = \begin{pmatrix} 1&0&0&0&0&0&1&0&1\\ 0&1&1&1&1&0&0&0&1\\ 0&0&1&1&1&1&1&1&1\\ 0&0&0&1&1&0&0&1&1\\ 1&1&1&1&1&1&1&1&1\\ 1&0&1&0&0&1&1&0&1\\ 1&1&1&0&0&1&1&1&1\\ 0&0&1&0&1&1&1&1&1\\ y_1&y_2&y_3&y_4&y_5&y_6&y_7&y_8&y_9 \end{pmatrix}$$

The encoder first checks W_1 and since $\mathrm{wt}(W_1) = 38 > 32$, it replaces W_1 with $\phi_1(W_1)$ (as highlighted in blue), where $\mathrm{wt}(\phi(W_1)) = 26$, and sets $y_1 = 1$.

$$A = \begin{pmatrix} 0&0&0&0&0&0&1&0&1\\ 0&1&1&0&0&0&0&0&1\\ 0&0&1&0&1&1&1&1&1\\ 0&0&0&1&1&0&0&0&1\\ 0&1&1&1&1&1&0&1&1\\ 0&0&1&0&0&1&1&0&1\\ 0&1&1&0&0&0&0&0&1\\ 0&0&1&0&1&1&1&1&1\\ 1&y_2&y_3&y_4&y_5&y_6&y_7&y_8&y_9 \end{pmatrix}$$

The encoder then checks W_2 and since $\mathrm{wt}(W_2) = 34 > 32$, it replaces W_2 with $\phi_1(W_2)$ (as highlighted in blue), where $\mathrm{wt}(\phi(W_2)) = 30$, and sets $y_3 = 1$.

$$A = \begin{pmatrix} 0&0&0&0&0&0&1&0&1\\ 0&1&1&0&0&0&0&0&1\\ 0&0&1&0&1&0&0&1&1\\ 0&0&0&1&1&0&0&0&0\\ 0&1&1&1&1&1&0&1&0\\ 0&0&1&0&0&1&1&0&1\\ 0&1&1&0&0&0&0&0&1\\ 0&0&1&0&1&1&1&1&1\\ 1&y_2&1&y_4&y_5&y_6&y_7&y_8&y_9 \end{pmatrix}$$

The encoder then checks C_1 and C_2 as two subarrays of size 8×7, and observes that their weights are both smaller than 28. It then sets $y_5 = y_7 = 0$. Since $y_1y_3y_5y_7 = 1100$, it sets $y_2y_4y_6y_8 = 0011$ and sets $y_9 = 0$. The final output is as follows.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Remark 4: To obtain capacity-approaching codes, the construction can be further extended for $m = n - k$ when k is no longer a constant. Since the redundancy of the encoder is $2(k+1)^2$, in order to get the asymptotic rate of the encoder to be one, it is sufficient to require $k = o(n)$.

B. SRT for $\mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon)$

In this subsection, we show that SRT is an efficient method to encode $\mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon)$. Similar to the results in Section IV, we show that for sufficiently large n, the coding method incurs at most one redundant bit. Recall the definition of (ℓ, ϵ)-r-forbidden and (ℓ, ϵ)-c-forbidden in Definition 7.

Lemma 3: Suppose that A is a binary array of size $n \times n$. If there is no (m, ϵ)-r-forbidden in any row of A or there is no (m, ϵ)-c-forbidden in any column of A then $A \in \mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon)$.

Proof: Clearly, if there is no (m, ϵ)-r-forbidden in any row of A, then every m consecutive bits in each row form an ϵ-balanced sequence. If we consider any window of size $m \times m$, since each row is ϵ-balanced, the window is also ϵ-balanced. ■

According to Lemma 3, one may view an array of size $n \times n$ as a binary sequence of length n^2 and then uses our SRT coding method (as presented in Section IV) to enforce the ϵ-balanced weight constraint over every m consecutive bits. We summarise the result as follows.

Theorem 9 (Modified Theorem 4): Given $n > 0$, $\epsilon \in (0, 1/2)$. For $(1/\epsilon^2)\ln(n^2) \leqslant m \leqslant n$, there exists linear-time algorithms $\textsc{Enc}_{\mathrm{Bal}_{\mathrm{SW}}(n,m;\epsilon)} : \{0,1\}^{n^2-1} \to \{0,1\}^{n^2}$ and $\textsc{Dec}_{\mathrm{Bal}_{\mathrm{SW}}(n,m;\epsilon)} : \mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon) \to \{0,1\}^{n^2-1}$ such that for all $\boldsymbol{x} \in \{0,1\}^{n^2-1}$ if $A = \textsc{Enc}_{\mathrm{Bal}_{\mathrm{SW}}(n,m;\epsilon)}(\boldsymbol{x})$, which is an array of size $n \times n$, then for every subsequence $\boldsymbol{w}$ consisting m consecutive bits of A_i, $\mathrm{wt}(\boldsymbol{w}) \in [(1/2 - \epsilon)m, (1/2 + \epsilon)m]$ for all $1 \leqslant i \leqslant n$. In other words, we have $A \in \mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon)$. Furthermore, we have $\textsc{Dec}_{\mathrm{Bal}_{\mathrm{SW}}(n,m;\epsilon)} \circ \textsc{Enc}_{\mathrm{Bal}_{\mathrm{SW}}(n,m;\epsilon)}(\boldsymbol{x}) \equiv \boldsymbol{x}$ for all $\boldsymbol{x} \in \{0,1\}^{n^2-1}$.

Remark 5: Given $\epsilon \in (0, 1/2)$, Theorem 9 requires $(1/\epsilon^2)\ln(n^2) \leqslant m \leqslant n$, or $(2/\epsilon^2)\ln n \leqslant m \leqslant n$. Since $n \gg \ln n$, Theorem 9 works for a wide range of m with respect to n.

VI. Conclusion

We have presented efficient encoding/decoding methods for two types of constraints over two-dimensional binary arrays: the p-bounded constraint and the ϵ-balanced constraint. The constraint is enforced over either every row and every column, regarded as the 2D row/column (RC) constrained codes, or over every window (where each window refers to as a subarray consisting of consecutive rows and consecutive columns), regarded as the 2D sliding-window constrained codes. The coding methods are based on: the divide and conquer algorithm and a modification of the Knuth's balancing technique, and the sequence replacement technique. Some encoding algorithms used the construction of antipodal matching as introduced in [11]. For certain code parameters, we have shown that there exist linear-time encoding/decoding algorithms that incur at most one redundant bit.

To conclude, we discuss open problems and possible future directions of research.

1) *Study the channel capacity.* The capacity of the constraint channels are defined by

$$\begin{aligned} \mathbf{c}_{\mathrm{RC}}(p) &\triangleq \lim_{n\to\infty} \frac{\log |\mathrm{B}_{\mathrm{RC}}(n; p)|}{n^2}, \\ \mathbf{c}_{\mathrm{RC}}(\epsilon) &\triangleq \lim_{n\to\infty} 1/n^2 \log |\mathrm{Bal}_{\mathrm{RC}}(n; \epsilon)|, \\ \mathbf{c}_{\mathrm{SW}}(m; p) &\triangleq \lim_{n\to\infty} \frac{\log |\mathrm{B}_{\mathrm{SW}}(n, m; p)|}{n^2}, \\ \mathbf{c}_{\mathrm{SW}}(m; \epsilon) &\triangleq \lim_{n\to\infty} 1/n^2 \log |\mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon)|. \end{aligned}$$

In this work we show that $\mathbf{c}_{\mathrm{RC}}(p) = 1$ for all $p \geqslant 1/2$, and $\mathbf{c}_{\mathrm{RC}}(\epsilon) = 1$ for all ϵ. On the other hand, the values $\mathbf{c}_{\mathrm{SW}}(m; p)$, $\mathbf{c}_{\mathrm{SW}}(m; \epsilon)$ remain unknown for fixed m, which is deferred to our future research work. Although we can design efficient encoders for $\mathrm{Bal}_{\mathrm{SW}}(n, m; p)$ when $m = n - o(n)$ or $\mathrm{Bal}_{\mathrm{SW}}(n, m; \epsilon)$ when $m \geqslant (2/\epsilon^2)\ln n$, a general construction for arbitrary values of m remains as an open problem.

2) *Combine the constrained encoders with error-correction capability.* To further reduce the error propagation during the decoding procedure, we are interested in the problem of combining our proposed encoders with error-correction capability. Recently, the problem of correcting multiple criss-cross deletions (or insertions) in arrays has been investigated in [31] and [32]. A natural question is whether such codes can be modified and adapted for our encoders so that the output arrays are 2D constrained codes that are also capable of correcting deletions, insertions, and substitutions.

References

[1] T. T. Nguyen, K. Cai, K. A. S. Immink, and Y. M. Chee, "Efficient design of capacity-approaching two-dimensional weight-constrained codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2930–2935.

[2] T. T. Nguyen, K. Cai, H. M. Kiah, K. A. S. Immink, and Y. M. Chee, "Using one redundant bit to construct two-dimensional almost-balanced codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 3091–3096.

[3] R. Talyansky, T. Etzion, and R. M. Roth, "Efficient code constructions for certain two-dimensional constraints," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 794–799, Mar. 1999.

[4] D. Psaltis, M. A. Neifeld, A. Yamamura, and S. Kobayashi, "Optical memory disks in optical information processing," *Appl. Opt.*, vol. 29, no. 14, pp. 2038–2057, 1990.

[5] J. J. Ashley, M. Blaum, and B. H. Marcus, "Report on coding techniques for holographic storage," IBM Res., New York, NY, USA, Tech. Rep. RJ 10013, 1996.

[6] D. Brady and D. Psaltis, "Control of volume holograms," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 9, no. 7, pp. 1167–1182, 1992.

[7] A. Chen, "Accessibility of nano-crossbar arrays of resistive switching devices," in *Proc. 11th IEEE Int. Conf. Nanotechnol.*, Aug. 2011, pp. 1767–1771.

[8] T. Raja and S. Mourad, "Digital logic implementation in memristor-based crossbars," in *Proc. Int. Conf. Commun., Circuits Syst.*, Jul. 2009, pp. 939–943.

[9] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014.

[10] R. B. Hur and S. Kvatinsky, "Memory processing unit for in-memory processing," in *Proc. IEEE/ACM Int. Symp. Nanosc. Architectures (NANOARCH)*, Jul. 2016, pp. 171–172.

[11] E. Ordentlich and R. M. Roth, "Low complexity two-dimensional weight-constrained codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Saint Petersburg, Russia, Jul. 2011, pp. 149–153.

[12] P. O. Vontobel, W. Robinett, P. J. Kuekes, D. R. Stewart, J. Straznicky, and R. S. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, Sep. 2009, Art. no. 425204.

[13] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Information-theoretic sneak-path mitigation in memristor crossbar arrays," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 4801–4813, Sep. 2016.

[14] X. Zhong, K. Cai, G. Song, and N. Raghavan, "Deep learning based detection for mitigating sneak path interference in resistive memory arrays," in *Proc. IEEE Int. Conf. Consum. Electron. Asia (ICCE-Asia)*, Seoul, South Korea, Nov. 2020, pp. 1–4.

[15] G. Song, K. Cai, X. Zhong, Y. Jiang, and J. Cheng, "Performance limit and coding schemes for resistive random-access memory channels," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2093–2106, Apr. 2021.

[16] X. Zhong, K. Cai, G. Song, W. Wang, and Y. Zhu, "Constrained coding and deep learning aided threshold detection for resistive memories," *IEEE Commun. Lett.*, vol. 26, no. 4, pp. 803–807, Apr. 2022.

[17] E. Ordentlich and R. M. Roth, "Two-dimensional weight-constrained codes through enumeration bounds," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1292–1301, Jul. 2000.

[18] A. Vardy, M. Blaum, P. H. Siegel, and G. T. Sincerbox, "Conservative arrays: Multidimensional modulation codes for holographic recording," *IEEE Trans. Inf. Theory*, vol. 42, pp. 227–230, Jan. 1996.

[19] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.

[20] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of some new efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 790–802, May 1996.

[21] E. Ordentlich, F. Parvaresh, and R. M. Roth, "Asymptotic enumeration of binary matrices with bounded row and column weights," in *Proc. IEEE Int. Symp. Inf. Theory*, Saint Petersburg, Russia, Jul. 2011, pp. 154–158.

[22] K. A. S. Immink and J. H. Weber, "Very efficient balanced codes," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 188–192, Feb. 2010.

[23] N. Alon, E. E. Bergmann, D. Coppersmith, and A. M. Odlyzko, "Balancing sets of vectors," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 1, pp. 128–130, Jan. 1988.

[24] T. T. Nguyen, K. Cai, and K. A. S. Immink, "Binary subblock energy-constrained codes: Knuth's balancing and sequence replacement techniques," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 37–41.

[25] T. T. Nguyen, K. Cai, K. A. S. Immink, and H. M. Kiah, "Capacity-approaching constrained codes with error correction for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 67, no. 8, pp. 5602–5613, Aug. 2021.

[26] C. D. Nguyen, V. K. Vu, and K. Cai, "Two-dimensional weight-constrained codes for crossbar resistive memory arrays," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1435–1438, May 2021.

[27] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.

[28] K. A. S. Immink, *Codes for Mass Data Storage Systems*, 2nd ed. Eindhoven, The Netherlands: Shannon Foundation Publishers, 2004.

[29] A. J. van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 200–207, Feb. 2010.

[30] O. Elishco, R. Gabrys, M. Medard, and E. Yaakobi, "Repeat-free codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 932–936.

[31] L. Welter, R. Bitar, A. Wachter-Zeh, and E. Yaakobi, "Criss-cross deletion correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 304–308.

[32] L. Welter, R. Bitar, A. Wachter-Zeh, and E. Yaakobi, "Multiple criss-cross deletion-correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2798–2803.

Tuan Thanh Nguyen received the B.Sc. and Ph.D. degrees in mathematics from Nanyang Technological University, Singapore, in 2014 and 2019, respectively. He was a Research Fellow with the School of Physical and Mathematical Sciences, Nanyang Technological University, from August 2018 to September 2019. He is currently a Research Fellow at the Advanced Coding and Signal Processing (ACSP) Laboratory, Singapore University of Technology and Design (SUTD). His research project concentrates on error correction codes and constrained codes for communication systems and data storage systems, especially codes for DNA-based data storage. His research interests include the interplay between combinatorics and computer science/engineering, particularly including combinatorics and coding theory.

Kui Cai (Senior Member, IEEE) received the B.E. degree in information and control engineering from Shanghai Jiao Tong University, Shanghai, China, the M.Eng. degree in electrical engineering from the National University of Singapore, and the joint Ph.D. degree in electrical engineering from the Technical University of Eindhoven, The Netherlands, and the National University of Singapore. She has been with the Data Storage Institute, Singapore, for 14 years. Currently, she is an Associate Professor with the Singapore University of Technology and Design (SUTD). Her main research interests are in the areas of coding theory, information theory, and signal processing for emerging data storage systems and digital communications. She received the 2008 IEEE Communications Society Best Paper Award in Coding and Signal Processing for Data Storage. She served as the Vice-Chair (Academia) for the IEEE Communications Society, Data Storage Technical Committee (DSTC) during 2015–2016. She was listed in the 2020 Who's Who in Engineering Singapore.

Han Mao Kiah (Senior Member, IEEE) received the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2014. From 2014 to 2015, he was a Post-Doctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign. From 2015 to 2018, he was a Lecturer with the School of Physical and Mathematical Sciences (SPMS), NTU, where he is currently an Assistant Professor. His research interests include DNA-based data storage, coding theory, enumerative combinatorics, and combinatorial design theory.

Kees A. Schouhamer Immink (Life Fellow, IEEE) received the Ph.D. degree (Hons.) from the University of Johannesburg in 2014. He founded Turing Machines Inc., an innovative start-up focused on novel signal processing for solid-state memories, where he is currently the President. He has been a Visiting Professor at the Singapore University of Technology and Design (SUTD). He has designed coding techniques of digital video, audio and data recording products, such as CD, DVD, and Blu-ray Disc. He was elected into the Royal Netherlands Academy of Sciences and the (U.S.) National Academy of Engineering. He received the Knighthood in 2000, the Personal Emmy Award in 2004, the 2017 IEEE Medal of Honor, the 1999 AES Gold Medal, the 2004 SMPTE Progress Medal, the 2014 Eduard Rhein Prize for Technology, the 2015 IET Faraday Medal, and the Golden Jubilee Award for Technological Innovation by the IEEE Information Theory Society in 1998.

Yeow Meng Chee (Senior Member, IEEE) received the B.Math. degree in computer science, and combinatorics and optimization and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1988, 1989, and 1996, respectively. Currently, he is a Professor of design and engineering with the National University of Singapore. Prior to this, he was a Professor of mathematical sciences with Nanyang Technological University, the Program Director of interactive digital media research and development with the Media Development Authority of Singapore, a Post-Doctoral Fellow with the University of Waterloo and the IBM's Zurich Research Laboratory, the General Manager of the Singapore Computer Emergency Response Team, and the Deputy Director of Strategic Programs at the Infocomm Development Authority, Singapore. His research interests include the interplay between combinatorics and computer science/engineering, particularly in combinatorial design theory, coding theory, extremal set systems, and their applications. He is a fellow of the Institute of Combinatorics and its Applications. He is an Editor of the *Journal of Combinatorial Theory, Series A*.